VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Kvalita a měření softwarových procesů

# Software Process Quality and Measurement

2019 Bc. Michal Přikryl

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Zadání diplomové práce

Student: **Bc. Michal Přikryl**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Kvalita a měření softwarových procesů
Software Process Quality and Measurement

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem této diplomové práce je vytvořit prototypový nástroj pro sledování a zobrazování metrik potřebných pro splnění standardu ASPICE zejména v nástroji pro záznam požadavků postavené na platformě Jazz od firmy IBM. Nástroj by měl umět definovat metriku a sledovat zadanou metriku v čase (trendové metriky), zobrazovat a generovat požadované reporty.

1. Analyzujte, jaké systémy pro metriky existují a případně jejich možnost propojení s IBM Jazz.
2. Analyzujte, jaké jsou možnosti rozšíření nalezených nástrojů pro požadované účely a zda lze nějaký vybraný nástroj použít a rozšířit.
3. Implementujte prototypový nástroj (za využití rozšíření stávajícího nalezeného nástroje nebo vlastní nástroj).
4. Vytvořte praktické ukázky, závěr a zhodnocení prototypového nástroje.

Seznam doporučené odborné literatury:

[1] John F. Sowa, Knowledge Representation: Logical, Philosophical, and Computational Foundations, Brooks Cole Publishing Co., Pacific Grove, CA, ©2000
[2] Alec Sharp, Patrick McDermott: Workflow Modeling: Tools for Process Improvement and Application Development, Artech House; 2 edition (October 31, 2008)
[3] Pfleeger, Shari Lawrence, and Joanne M. Atlee. 2009. Software Engineering: Theory and Practice: Prentice Hall, ISBN 0136061699
[4] Pressman, Roger S. 2010. Software Engineering : A Practitioner's Approach. 7th ed. New York: McGraw-Hill Higher Education, ISBN 9780073375977
[5] Sommerville, Ian. 2010. Software Engineering. 9th ed, International Computer Science Series. Harlow: Addison-Wesley, ISBN 978-0137035151
[6] Automotive SPICE® Process Reference and Assessment Model (PDF 1800KB) - RELEASE 3.1 - 01 November 2017, http://www.automotivespice.com/download/
Další literatura podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Svatopluk Štolfa, Ph.D.**

Datum zadání: 01.09.2018

Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
*vedoucí katedry*

prof. Ing. Pavel Brandštetter, CSc.
*děkan fakulty*

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 25. dubna 2019 .....

Rád bych poděkoval Ing. Svatopluku Štolfovi, Ph.D. za vedení této diplomové práce, odbornou pomoc a věcné rady a připomínky při tvorbě této diplomové práce. Dále panu Ing. Zdeňku Velartovi, Ph.D. za pomoc při instalaci a zprovoznění platformy IBM Jazz a za praktické rady a připomínky při implementaci nástroje.

**Abstrakt**

Cílem této diplomové práce je studium metrik softwarového vývoje a vytvoření prototypového nástroje pro sledování a zobrazování metrik nutných pro splnění standardu Automotive SPICE. První část práce je věnována studiu problematiky měření kvality a metrik, jak z pohledu obecného, tak v rámci standardu Automotive SPICE, dále analýze stávajících řešení a možností jejich využití nebo rozšíření ve vyvíjeném nástroji. Zvláštní důraz je kladen na analýzu nástroje IBM DOORS Next Generation, který bude v praktické části této práce rozšiřován vyvíjeným nástrojem. Druhá část práce se tedy zabývá implementací prototypového nástroje pro sledování a zobrazování metrik, spojených s disciplínou sběru požadavků, v rámci softwarového procesu řízeného standardem Automotive SPICE.

**Klíčová slova**: diplomová práce, metriky a měření kvality, softwarový proces, IBM Jazz, IBM DOORS Next Generation, XML, Automotive SPICE, C#, .NET Core, JavaScript, HTML

**Abstract**

The goal of this thesis is a study of software process measurement and metrics and the creation of a prototype tool for monitoring and displaying metrics, which are necessary to meet the Automotive SPICE standard. The first part of this thesis is devoted to the study of quality measurement and metrics, both from a general point of view and within Automotive SPICE standard, as well as an analysis of existing solutions and possibilities of their utilization or expansion in the developed tool. Particular emphasis is placed on the analysis of the IBM DOORS Next Generation tool, which will be extended in the practical part of this thesis by the developed tool. The second part of this thesis deals with the implementation of a prototype tool for monitoring and displaying metrics, related to the discipline of collecting requirements, within software process controlled by Automotive SPICE standard.

**Key Words**: master thesis, metrics and quality measurement, software process, IBM Jazz, IBM DOORS Next Generation, XML, Automotive SPICE, C#, .NET Core, JavaScript, HTML

# Obsah

# Seznam použitých zkratek a symbolů

| | | |
|---|---|---|
| ASP.NET | – | Application Lifecycle Management |
| ASPICE | – | Automotive Software Process Improvement and Capability dEtermination |
| ASP.NET | – | Active Server Pages .NET |
| BPMN | – | Business Process Model and Notation |
| cbQL | – | Code Beamer Query Language |
| CFP | – | Call For Proposals |
| CORS | – | Cross-Origin Requests |
| CSV | – | Comma-Separated Values |
| DDL | – | Data Definition Language |
| DevOps | – | Development and Operations |
| DNG | – | DOORS Next Generation |
| DML | – | Data Manipulation Language |
| DOORS | – | Rational Dynamic Object Oriented Requirements System |
| EA | – | Enterprise Architect |
| HTML | – | Hyper Text Markup Language |
| HTTP | – | Hypertext Transfer Protocol |
| HTTPS | – | Hypertext Transfer Protocol Secure |
| IEC | – | International Electrotechnical Commission |
| IIS | – | Internet Information Services |
| ISO | – | International Organization for Standardization |
| ITT | – | Invitation To Tender |
| JSON | – | JavaScript Object Notation |
| JWT | – | JSON Web Token |
| LDAP | – | Lightweight Directory Access Protocol |
| LINQ | – | Language Integrated Query |
| MVC | – | Model View Controller |
| MDG | – | Model Driven Generation |
| MS | – | Microsoft |
| OWL | – | Web Ontology Language |
| OS | – | Operační Systém |
| OSLC | – | Open Services for Lifecycle Colaboration |
| PA | – | Process attribute |
| PDF | – | Portable Document Format |
| PNG | – | Portable Network Graphics |
| QA | – | Quality Assurance |

| | | |
|---|---|---|
| RDF | – | Resource Description Framework |
| ReqIF | – | Requirements Interchange Format |
| REST API | – | Representational State Transfer Application Programming Interface |
| SaaS | – | Software as a Service |
| SPICE | – | Software Process Improvement and Capability dEtermination |
| SQL | – | Structured Query Language |
| SSL | – | Secure Sockets Layer |
| SysML | – | Systems Modeling Language |
| UML | – | Unified Modeling Language |
| W3C | – | World Wide Web Consortium |
| WebAPI | – | Web Application Programming Interface |
| WSDL | – | Web Services Description Language |
| XML | – | Extensible Markup Language |

# Seznam obrázků

# Seznam tabulek

# Seznam výpisů zdrojového kódu

# 1 Úvod

S rostoucí velikostí a složitostí softwaru roste také náročnost jeho vývoje, to klade i v době agilních metod stále větší důraz na korektně a důkladně provedený softwarový proces. K tomu, aby byl software vyvíjen s určitým řádem, a nejen náhodně právě dopomáhá dodržování určitého softwarového procesu. Ian Sommerville definuje softwarový proces jako množinu aktivit, vedoucích k vytvoření softwarového produktu a také jako proces, poskytující standardizovaný formát pro plánování, organizování a průběh vývoje [1].

Dodržování softwarového procesu sebou přináší i další výhody, jednou z nich je i nemalá úspora nákladů, jelikož je například možné jednotlivé podprocesy opakovat v rámci různých projektů, čímž se mimo jiné i urychlí vývoj projektu.

Existují různé nástroje, metodiky a standardy podporující průběh vývoje softwaru. Nástroje usnadňují například správu softwarového procesu, ukládání artefaktů, dokumentaci a celkovou administraci projektu. Metodiky doporučují, jak správně vyvíjet a postupovat při vývoji software. Nakonec standardy předepisují, jak by měl kompletní vývojový proces vypadat, jaké podprocesy by se měly provádět a co by mělo být jejich korektním výsledkem. Příkladem takového standardu může být Automotive SPICE, o kterém je blíže pojednáváno v kapitole 2.1.

Všechny informace získané v průběhu vývoje je vhodné dále zaznamenávat a zpracovávat. Proto je důležitým aspektem profesionálního vývoje softwaru kvalita a měření kvality softwaru, potažmo softwarového procesu. Díky metrikám lze získat přehlednější pohled na plnění podmínek řízeného procesu. Měření a s ním spojené metriky prochází celým vývojovým procesem od počátku projektu, až po jeho zdárné ukončení. Kvalitu softwaru lze měřit například testováním, zatímco kvalitu softwarového procesu lze lépe měřit metrikami. O metrikách a měření softwarového procesu je pojednáváno v kapitole 3.

Softwarový vývoj by byl složitý bez podpůrných nástrojů, které pomáhají se zvládnutím mnoha rozličných úkolů. Množství existujících nástrojů a aplikací se zaměřuje na podporu vývoje a průběhu softwarového procesu. Mezi těmito nástroji lze najít větší či menší rozdíly v pokrytí všech kroků softwarového procesu, dále také v celkové pokročilosti nástroje nebo kvalitě uživatelského rozhraní. Analýzou a srovnáním vybraných, již existujících, aplikací a nástrojů se zabývá kapitola 5. V kapitole 4 je detailněji popsána platforma IBM Jazz, potažmo její nástroj IBM DOORS Next Generation, jelikož ten je v hojné míře využívána profesionály z oboru a také je i součástí implementace prototypového nástroje.

Součástí této práce je také prototypový nástroj, popsán v kapitole 6, pro definici a sledování zadaných metrik v čase a v aktuálním stavu. Implementovaný nástroj blíže spolupracuje s platformou IBM Jazz, konkrétně s jejím nástrojem IBM DOORS Next Generation, jehož funkcionalitu v oblasti metrik a měření kvality rozšiřuje natolik, aby byla umožněna jednoduchá a automatizovaná kontrola dodržování zásad vývoje nutných ke splnění standardu Automotive SPICE.

# 2 Standardy a zlepšení procesu

V dnešní době je stále větší poptávka po kvalitním softwaru za nízkou cenu. Proto se mnoho softwarových společností zaměřilo na zlepšení vývojového procesu, jehož požadovaným výsledkem je zvýšení kvality softwaru, snížení celkových nákladů a urychlení vývoje systému. Procesní zlepšení znamená porozumět existujícím procesům a změnit tyto procesy tak, aby bylo jejich výsledkem minimálně zvýšení kvality softwaru.

Dle Iana Sommervilla existují dva zcela odlišné přístupy ke zlepšení procesů:

- *Přístup procesní vyspělosti*, který se zaměřuje na zlepšení procesů a správy projektu, a přináší správné praktiky softwarového inženýrství do projektového týmu nebo společnosti. Úroveň procesní vyspělosti reflektuje míru, jak dobré technické a manažerské praktiky byly použity v procesu vývoje software. Primárním cílem tohoto přístupu je zvýšení kvality produktu a předvídatelnosti procesu [2].

- *Agilní přístup*, který se zaměřuje na iterativní vývoj a redukci režijních nákladů softwarového procesu. Charakteristickými vlastnostmi agilních metod jsou rychlé poskytování funkčnosti a schopnost reagovat na požadavky zákazníka [2].

Každý z přístupů má své výhody a nevýhody. Přístup procesní vyspělosti je založen na plánovaném vývoji a často vyžaduje zvýšení režijních nákladů ve smyslu, že prováděné aktivity se často netýkají programování – primární výdělečné činnosti. Agilní přístup se zaměřuje na vyvíjený kód a minimalizuje dokumentaci. Z tohoto důvodu Ian Sommerville doporučuje využít agilní přístup spíše v malých společnostech a týmech, kde tento přístup zajistí větší zlepšení procesu. Zatímco pro větší společnosti nebo složitější a kritické projekty doporučuje využít přístupu procesní vyspělosti [2].

Metriky a měření kvality softwaru spíše patří k přístupu procesní vyspělosti, proto bude tato práce věnována převážně tomuto přístupu. Proces zlepšení procesu je cyklický a skládá se ze čtyř podprocesů. Prvním podprocesem je proces měření, kdy probíhá měření atributů aktuálního procesu a dle výsledků těchto měření se vyhodnocuje, zda je prováděný proces efektivní. Druhým podprocesem je proces analýzy, ve kterém probíhá posouzení aktuálního procesu a jsou identifikována slabá místa a nedostatky procesu. Třetím podprocesem je proces změny, kdy dochází ke změnám procesu v místech, kde docházelo k nedostatkům.

Jako pomoc při posuzování procesní vyspělosti a úrovně se používají standardy. Jedním z takových standardů je SPICE, který definuje množinu požadavků pro posouzení procesu. Záměrem standardu SPICE je pomoci organizacím při vytváření objektivního hodnocení jakýchkoli definovaných procesů [3]. Rozšířením standardu SPICE je standard Automotive SPICE, o kterém pojednává následující podkapitola 2.1.

## 2.1 Automotive SPICE

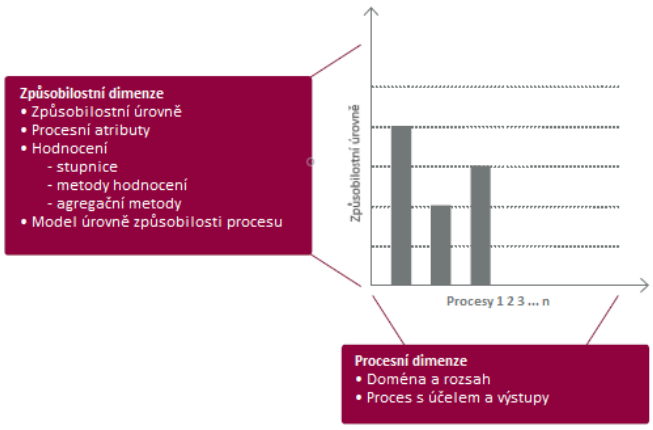
Automotive SPICE (ISO15504) je standard, využívaný jako framework pro řízení procesů a kvality software, převážně ve sféře automobilového průmyslu. Standard Automotive SPICE zahrnuje nejnovější poznatky a zkušenosti s vedením projektu v automobilovém průmyslu. Byl vyvinut a je udržován společnostmi jako Audi, BMW, Daimler a mnoha dalšími. Aktuální verzí je verze 3.1 z listopadu 2017 [4].

Obrázek 1: Logo Automotive SPICE

Tento standard se využívá při vývoji mechatronických systémů, kde se převážně zaměřuje na domény softwaru a systému produktu. Nově od aktuální verze 3.1 je také možné zaměření na hardwarovou a mechanickou část produktu. Splnění standardu Automotive SPICE je nutnou prerekvizitou pro možnost ucházet se o dodání softwarových děl předním (nejen) evropským automobilovým výrobcům.

Díky dodržování standardu Automotive SPICE lze efektivně vyvíjet software, což dále zahrnuje zajištění znovupoužitelnosti jednotlivých komponent softwarového díla, vznik podrobné dokumentace celého vývoje, kvalitní otestování software a také v neposlední řadě lze díky předem známým postupům lépe předpovídat dobu, nutnou k vývoji. Všemi těmito kroky lze dosáhnout vysoké kvality vývoje softwaru a tím i získání prestižní zakázky.

Koncept určení způsobilosti procesu při použití Automotive SPICE je založen na dvoudimenzionálním rámci (viz obrázek 2), který se skládá z procesní dimenze, složené z procesů definovaných v procesním referenčním modelu, a ze způsobilostní dimenze, tvořené úrovněmi způsobilosti, které jsou dále rozděleny na procesní atributy. Model hodnocení procesu vybírá procesy z procesního referenčního modelu a doplní je indikátory, které podporují rozhodování hodnotitele při určování hodnocení procesu vzhledem k dimenzi způsobilosti.




Obrázek 2: Model hodnocení procesu při použití Automotive SPICE [5]

## 2.2 Framework pro měření úrovně způsobilosti

Dimenze způsobilosti se skládá z úrovní způsobilosti, které hodnotí proces na základě splnění procesních atributů. Tyto procesní atributy zajišťují měřitelnou charakteristiku pro měření úrovně způsobilosti. Framework pro měření poskytuje nezbytné požadavky a pravidla pro určení dimenze způsobilosti. Taktéž definuje schéma, které umožňuje hodnotiteli rozhodnout úroveň způsobilosti daného procesu. Tyto úrovně způsobilostí jsou součástí frameworku pro měření.

Aby bylo možné hodnotit, musí každý proces obsahovat atributy, které je nutné dodržet pro získání určité úrovně způsobilosti. Automotive SPICE používá framework pro měření definovaný ve standardu ISO/EIC 332020:2015.

Úroveň způsobilosti je tedy množina procesních atributů, které v celku tvoří významné vylepšení schopnosti provádět proces/y. Podle ISO/EIC 332020:2015 existuje šest úrovní způsobilosti, zahrnujících devět procesních atributů (zkratka PA), viz tabulka 1.

| Úroveň | Popis úrovně |
|---|---|
| **0 - nekompletní** | Proces není implementován nebo nedosáhne svého cíle. |
| **1 - vykonávaný** | Proces je implementován a dosáhne svého cíle.<br>**PA.1.1** – atribut výkonnosti procesu |
| **2 - řízený** | Proces je implementován v řízeném prostředí a jeho produkty jsou vhodně zavedeny, řízeny a udržovány.<br>**PA.2.1** – atribut managementu výkonu<br>**PA.2.2** – atribut managementu pracovních produktů |
| **3 - zavedený** | Proces je implementován za použití standardního procesu, který je schopný dosáhnout svých výsledků.<br>**PA.3.1** – atribut definice procesu<br>**PA.3.2** – atribut procesního nasazení |
| **4 - předvídatelný** | Proces je předvídatelný v rámci definovaných limitů, vedoucích k dosažení procesních výstupů.<br>**PA.4.1** – atribut kvantitativní analýzy<br>**PA.4.2** – atribut kvantitativní kontroly |
| **5 - optimalizovaný** | Proces je kontinuálně vylepšován, aby reagoval na změny v souladu s cíli organizace.<br>**PA.5.1** – atribut procesní inovace<br>**PA.5.2** – atribut implementace procesní inovace |

Tabulka 1: Úrovně způsobilosti procesu

### 2.2.1 Hodnocení procesních atributů

Pro podporu hodnocení procesních atributů nabízí framework stupnici hodnocení přehledně vypsanou v tabulce 2. Jednotlivé procesní atributy jsou posuzovány na základě naplnění podmínek daných procesem.

| | |
|---|---|
| **N - nedosaženo** | Existuje pouze malý nebo žádný důkaz o dosažení daného procesního atributu. **0 až 15%** dosažení |
| **P - částečně dosaženo** | Existují určité důkazy o přístupu a částečném dosažení atributu v posuzovaném procesu. **16% až 50%** dosažení |
| **L - téměř dosaženo** | Existují důkazy o systematickém přístupu a významném dosažení atributu v posuzovaném procesu. **51% až 85%** dosažení |
| **F - plně dosaženo** | Existují důkazy o úplném a systematickém přístupu a plném dosažení atributu v posuzovaném procesu. **86% až 100%** dosažení |

Tabulka 2: Stupnice hodnocení dle ISO/IEC 33020

Stupně hodnocení P a L mohou být dále rozděleny na P- a P+, respektive L- a L+. Kdy úroveň P- nebo L- znamená částečné plnění základní úrovně P nebo L, ale ne tak nízké, aby se hodnocení snížilo o celou úroveň. Úroveň P+ a L+ znamená mírně nadstandardní plnění úrovně P nebo L, ale ne natolik, aby splňovalo požadavky vyšší úrovně L nebo F.

### 2.2.2 Model úrovně způsobilosti procesu

Úroveň způsobilosti procesu dosažená procesem může být odvozena z hodnocení procesních atributů tohoto procesu. Model úrovně způsobilosti procesu definuje pravidla, jak dosažení určité úrovně závisí na hodnocení procesních atributů pro posuzovanou úroveň a všechny nižší úrovně. Základním pravidlem je, že dosažení dané způsobilostní úrovně vyžaduje stupeň hodnocení L (téměř dosaženo) úrovní odpovídajících procesních atributů a stupeň hodnocení F (plně dosaženo) všech procesních atributů nižších způsobilostních úrovní.

## 2.3 Model hodnocení procesu

Model hodnocení procesu nabízí ukazatele pro zjištění, zda výstupy procesu a procesních atributů v započatých procesech projektu a organizačních jednotek existují nebo chybí. Tyto ukazatele pomáhají hodnotitelům v rozhodování o míře způsobilosti. Existují dva typy ukazatelů – ukazatele výkonnosti procesu a ukazatele způsobilosti procesu.

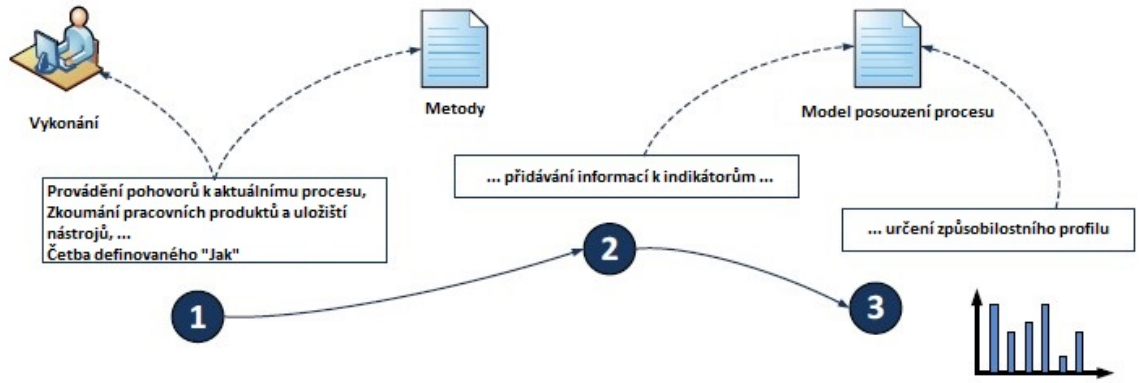
Ukazatele hodnocení se používají k potvrzení, že určité praktiky byly provedeny, jak dokládají podklady, získané během hodnotícího procesu. Podklady jsou získány z kontroly pracovních produktů posuzovaného procesu a z informací získaných od manažerů procesu.

### 2.3.1 Ukazatele výkonnosti procesu

Ukazatele výkonnosti procesu se týkají výlučně první úrovně způsobilosti. Typy těchto ukazatelů jsou základní postupy a pracovní produkty, oba tyto typy jsou procesními výstupy. Základní postupy představují ukazatele orientované na činnost a pracovní produkty představují ukazatele orientované na výsledek. Nejedná se o generické ukazatele, nýbrž jsou pro každý proces unikátní. Existence základních postupů a pracovních produktů poskytuje podklady o výkonnosti procesů spojených s nimi.

### 2.3.2 Ukazatele způsobilosti procesu

Ukazatele způsobilosti procesu se vztahují na druhou až pátou úroveň způsobilosti. Existují dva typy ukazatelů způsobilosti procesu – obecný postup a obecný zdroj. Oba typy se týkají jednoho nebo více procesních atributů. Ve srovnání s výkonnostními ukazateli jsou obecného typu, je tedy možné je například použít na jakýkoli proces.




Obrázek 3: Postup hodnocení procesní způsobilosti při použití Automotive SPICE [6]

## 2.4 Procesní dimenze a procesní referenční model

Procesní referenční model obsahuje jednotlivé procesy. Každý proces z procesní dimenze se skládá z procesního referenčního modelu a ukazatelů výkonnosti procesu nutných k definici modelu hodnocení procesu. V rámci ukazatelů výkonnosti procesu jsou vždy definovány obecné postupy a konkrétní výstupy (pracovní produkty), které představují očekávané pozitivní výsledky procesu.

Procesy jsou rozděleny do tří skupin. Těmito skupinami jsou primární procesy, podpůrné procesy a organizační procesy. V každé skupině jsou dále procesy rozděleny do skupin podle toho, jakým typem činnosti se zabývají. Jednotlivé procesy jsou identifikovatelné pomocí krátkého označení, kde první tři znaky určují skupinu, do které patří, poté následuje separátor – tečka, a jako poslední znak, případně dva znaky, je identifikační číslo procesu v rámci dané skupiny.

### 2.4.1 Primární procesy

Kategorie primárních procesů obsahuje procesy, které mohou být využity zákazníkem při získávání smluvního produktu od dodavatele a dodavatelem při dodávání produktů zákazníkům. Skládá se ze čtyř skupin - akvizičních procesů, dodavatelských procesů, procesů systémového inženýrství a procesů softwarového inženýrství.

**2.4.1.1 Akviziční procesy (ACQ)** Skupina akvizičních procesů se skládá z procesů, které jsou prováděny zákazníkem nebo dodavatelem, který vystupuje jako zákazník pro své dodavatele, v obou případech za účelem získání produktu nebo služby.

- ACQ.3 – Proces dohody o smlouvě, kdy dochází k vyjednání a schválení smlouvy s dodavatelem, výsledkem tohoto procesu je kompletní smlouva, se kterou souhlasí obě smluvní strany.
- ACQ.4 – Proces monitorování dodavatele, kdy se kontroluje plnění předem stanovených požadavků a v případě nutnosti se domlouvají změny smlouvy.
- ACQ.11 – Proces technických požadavků, kde jsou stanoveny technické požadavky, což zahrnuje získávání funkčních a nefunkčních požadavků.
- ACQ.12 – Proces právních a administrativních požadavků, který se zabývá výkladem práva a právních závazků, které jsou v souladu s národními a mezinárodními zákony.
- ACQ.13 – Proces projektových požadavků specifikuje požadavky pro ujištění, že je projekt zpracováván s adekvátním plánováním, personálním obsazením a je také organizací spravován dle předem daných aktivit a úkolů.
- ACQ.14 – Proces s názvem požadavek na návrhy, slouží k přípravě a vydání potřebných akvizičních požadavků. Sbírá informace pro dokumenty CFP a ITT.

- ACQ.15 – Posledním procesem je proces dodavatelské kvalifikace, kde je kontrolována způsobilost dodavatele, zda má požadovanou kvalifikaci pro účast ve výběrovém řízení, jako například technické zázemí, uživatelskou podporu atp.

#### 2.4.1.2 Dodavatelské procesy (SPL)

Skupina dodavatelských procesů obsahuje procesy, které jsou prováděny dodavatelem za účelem dodání produktu nebo služby.

- SPL.1 – Proces výběrového řízení pro dodavatele, jehož smyslem je příprava na zákazníkovi požadavky, výběrové řízení, podání nabídky do výběrového řízení a potvrzení přidělení smlouvy.
- SPL.2 – Proces uvolnění produktu, jehož účelem je kontrolované uvolnění/předání produktu daným zákazníkům. Definuje podprocesy, které je nutné splnit pro úspěšné předání produktu, jakým je například vytvoření předávací dokumentace.

#### 2.4.1.3 Procesy systémového inženýrství (SYS)

Skupina procesů systémového inženýrství je složena z procesů, řešících získání a správu zákaznických a interních požadavků, definuje architekturu, integraci a testování na systémové úrovni.

- SYS.1 – Proces elicitace požadavků, týkající se získání požadavků a sledování jejich dalšího vývoje v průběhu životního cyklu produktu nebo služby. V rámci tohoto procesu je také stanovena bázová linie požadavků, která slouží jako základ pro pracovní produkty.
- SYS.2 – Proces analýzy zákaznických požadavků transformuje definované požadavky účastníka (zákazníka) do systémových požadavků, které budou ovlivňovat návrh systému.
- SYS.3 – V procesu s názvem návrh systémové architektury dochází k vytvoření návrhu architektury systému a propojení systémových požadavků na konkrétní části systému.
- SYS.4 – Proces systémové integrace a integračního testování slouží k integraci částí systému, aby byl výsledkem konzistentní a kompletně otestovaný systém se správnou systémovou architekturou.
- SYS.5 – Proces testování kvalifikace systému kontroluje, zda integrovaný systém splňuje systémové požadavky a zda je připraven pro předání zákazníkovi.

#### 2.4.1.4 Procesy softwarového inženýrství (SWE)

Procesy softwarového inženýrství zahrnují procesy, zabývající se správou softwarových požadavků získaných z požadavků systémových, dále o vývoj softwarové architektury a návrhu a v neposlední řadě také o implementaci, integraci a testování.

- SWE.1 – Proces analýzy softwarových požadavků, kde dochází k transformaci systémových požadavků na softwarové požadavky.

- SWE.2 – Proces návrhu softwarové architektury má za cíl vytvořit architektonický návrh a určit, které softwarové požadavky patří k určité části softwaru.
- SWE.3 – Proces podrobného softwarového návrhu a konstrukce softwarových jednotek – unit, poskytuje podrobný návrh softwarových komponent a specifikuje a vytváří softwarové jednotky.
- SWE.4 – Proces verifikace softwarových jednotek zajišťuje, jak už název napovídá, verifikaci softwarových jednotek k zajištění důkazu o shodě s podrobným softwarovým návrhem a s nefunkčními softwarovými požadavky.
- SWE.5 – Proces softwarové integrace a integračního testování zahrnuje integraci softwarových jednotek do větších softwarových prvků až do kompletního, konzistentního a otestovaného celku.
- SWE.6 – Proces softwarových kvalifikačních testů požaduje, aby byl integrovaný software otestovaný pro zajištění souladu se softwarovými požadavky.

### 2.4.2 Podpůrné procesy (SUP)

Kategorie podpůrných procesů je složena z procesů, které mohou být použity libovolným z ostatních procesů v různých částech životního cyklu.

- SUP.1 – Proces zajištění kvality poskytuje záruku, že pracovní produkty a procesy jsou v souladu s předem stanovenými ustanoveními a plány, a že veškeré nesrovnalosti jsou vyřešeny.
- SUP.2 – Proces verifikace potvrzuje, že každý pracovní produkt procesu nebo projektu reflektuje specifikované požadavky.
- SUP.4 – Proces společné revize udržuje společné porozumění mezi zainteresovanými stranami vzhledem k cílům dohody a definuje, co musí být provedeno, aby bylo možné zajistit vývoj produktu, který uspokojuje zúčastněné strany.
- SUP.7 – Proces dokumentace má za úkol vytvoření a údržbu zaznamenaných informací, které produkují procesy.
- SUP.8 – Proces konfiguračního managementu se zabývá vytvářením a údržbou integrity všech pracovních produktů procesu nebo projektu.
- SUP.9 – Proces managementu řešení problémů zajišťuje, že problémy jsou identifikované, analyzované, spravované a řízené k vyřešení.
- SUP.10 – Účelem procesu managementu požadavků na změny je ujištění, že všechny změny jsou řízeny, sledovány a implementovány.

### 2.4.3 Organizační procesy

Kategorie organizačních procesů se skládá z procesů vyvíjejících procesní, produktové a zdrojové prostředky, které mohou při správném použití v projektech organizaci pomoct dosáhnout kýžených obchodních cílů.

**2.4.3.1 Procesy managementu (MAN)** Skupina procesů managementu obsahuje procesy, které mohou být využity kýmkoli, kdo spravuje projekt jakéhokoli typu nebo také proces v rámci životního cyklu projektu.

- MAN.3 – Proces projektového managementu identifikuje, vytváří a spravuje aktivity a zdroje projektu nutné k vytvoření produktu v kontextu projektových požadavků a omezení.
- MAN.5 – Proces managementu rizik průběžně identifikuje, analyzuje, opravuje a monitoruje rizika.
- MAN.6 – Proces měření sbírá a analyzuje data týkající se produktů a procesů implementovaných v rámci organizace a jejích projektů, dále podporuje efektivní řízení procesů a objektivně demonstruje kvalitu produktu.

**2.4.3.2 Procesy zlepšení procesů (PIM)** Skupina procesů, s názvem zlepšení procesu, se skládá pouze z jednoho procesu. Tento proces obsahuje praktiky, vedoucí ke zlepšení proveditelnosti procesů v rámci organizační jednotky.

- PIM.3 – Účelem procesu zlepšení procesů je kontinuální zlepšování efektivity organizace skrze používané procesy a přizpůsobení se potřebám podniku.

**2.4.3.3 Procesy znovupoužitelnosti (REU)** Skupina procesů znovupoužitelnosti se také skládá pouze z jednoho procesu, který systematicky využívá možnosti opětovného využití programu v rámci organizace.

- REU.2 – Účelem procesu opětovného použití programu je plánovat, vytvořit, spravovat, kontrolovat a monitorovat znovupoužití programu v rámci organizace a systematicky využívat možnosti opětovného využití programů.

# 3 Metriky a měření kvality software

Aby bylo možné vytvořit nástroj pro sledování a zobrazování metrik je nutné se více seznámit s obecnou teorií metrik, tudíž bude tato kapitola zaměřena na jejich problematiku, využití a stále rostoucí význam při efektivním zavádění a prováděné procesů. V kontextu této práce budou probírány a využity metriky spojené převážně se softwarovým procesem a vývojem, konkrétně metriky v oblasti sběru požadavků a obecně se týkající požadavků.

## 3.1 Úvod do metrik a měření kvality

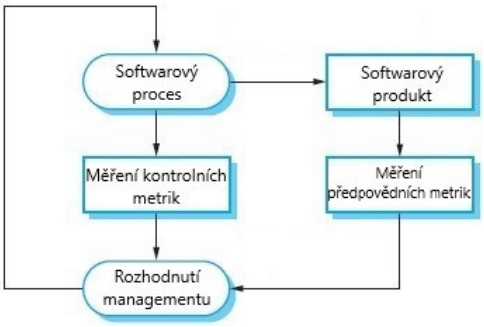
Při kvalitním vývojovém procesu je nutné dodržovat množství „best practices“, ty jsou definovány standardy, jako například Automotive SPICE (o kterém je pojednáváno v kapitole 2.1). Zavádění sofistikovaných praktik do vývojového procesu je znakem zlepšování kvality vývoje. Mezi takové praktiky patří vytváření podrobné dokumentace, pravidelná komunikace s klientem, dodržování stanovených nároků a požadavků na kvalitu a mnoho dalších. Dodržování těchto praktik lze mimo jiné posuzovat pomocí metrik a měření kvality.

Metriky umožňují monitorovat, vyhodnocovat a porovnávat průběh procesu, jak už softwarového, tak i jakéhokoli jiného. Metriky jsou výstupem, který lze při správném použití využít i ke vylepšení zavedených procesů, což může mít za následek například zjednodušení a optimalizaci daného procesu. Ian Sommerville [7] popisuje softwarové měření jako činnost, která se týká vyvození číselné hodnoty atributu softwarové komponenty, systému nebo procesu. Při porovnávání získaných hodnot s referencí získanou například ze standardu aplikovaného společností, lze díky měření a metrikám získat podrobný přehled o kvalitě vyvíjeného softwaru nebo posoudit kvalitu a efektivitu softwarového procesu. Metrika je charakteristikou softwarového systému, systémové dokumentace nebo vývojového procesu, kterou lze objektivně měřit. V dlouhodobém časovém měřítku lze využít metriky jako rozhodovací prvek v případě posouzení trendu poklesu nebo nárůstu kvality software, jelikož nabízí jednodušší pohled na kvalitu než například revize.

Elementárním příkladem metrik mohou být metriky, se kterými se dostala do kontaktu většina vývojářů – velikost produktu měřená v počtu řádků kódu nebo počet reportovaných chyb v softwarovém produktu v čase nebo počet dní nutných k vývoji softwarové komponenty.

Ian Sommerville [1] definuje dvě základní kategorie softwarových metrik a to kontrolní metriky a předpovědní metriky. Jak názvy napovídají, kontrolní metriky se týkají správy procesů a jsou tedy často spojeny se softwarovým procesem, zatímco předpovědní metriky předpovídají vlastnosti a charakteristiky softwaru. Příkladem kontrolní metriky může být průměrné úsilí a čas potřebný k vyřešení reportované chyby v softwaru. Předpovědní metriky jsou spíše spjaty se samotným softwarem, tudíž se jim často také říká produktové metriky. Příkladem produktové metriky může být cyklomatická složitost modulu softwaru nebo průměrná délka identifikátoru proměnné ve zdrojovém kódu. Obrázek 4 znázorňuje, jak oba druhy metrik ovlivňují rozhodování managementu. Manažeři používají procesní (kontrolní) měření k rozhodnutí, jestli mají

být změny provedeny, a předpovědní měření ke zjištění, kolik úsilí a zdrojů je potřeba využít k provedení změn softwaru.




Obrázek 4: Kontrolní a předpovědní měření [8]

## 3.2 Produktové metriky

Produktové metriky jsou předpovědní metriky, které se využívají pro měření interních atributů softwarového systému. Příkladem mohou být systémová velikost měřená v počtu řádků kódu nebo počet metod spojených s objektem třídy. Produktové metriky lze dále dělit na dva typy a to na dynamické metriky a statické metriky.

- *Dynamické metriky* – jsou vytvářeny za běhu programu. Mohou být sbírány během systémového testování nebo tehdy, když systém přejde do produkce. Příkladem je počet chybových reportů nebo čas nutný k dokončení výpočtu.
- *Statické metriky* – jsou sbírány měřením prováděným nad reprezentací systému, jakou je návrh, program nebo dokumentace. Příkladem statické metriky je velikost kódu nebo průměrná délka použitých identifikátorů proměnných.

Tyto typy metrik se vážou k různým kvalitativním atributům. Dynamické metriky pomáhají ohodnotit efektivitu a spolehlivost programu, zatímco statické metriky pomáhají při hodnocení složitosti, srozumitelnosti a udržovatelnosti softwarového systému nebo softwarové komponenty.

## 3.3 Kontrolní metriky

Kontrolní metriky, jinými slovy procesní metriky, jsou kvantitativní údaje o softwarovém procesu, jako například čas potřebný k provedení nějakého procesního úkonu, kdy pod procesním úkonem si lze představit vytváření testovacího případu. Procesní metriky jsou základem pro zlepšení provádění procesů, jelikož z nich lze vyvodit znaky poklesu či nárůstu kvality provádění procesu.

Existují 3 druhy procesních metrik:

- *Čas nutný k dokončení určitého procesu* – jde například o celkový čas věnovaný procesu, čas konkrétních inženýrů strávený na procesu a další.

- *Zdroje potřebné pro konkrétní proces* – zdroje mohou obsahovat celkové úsilí v počtu vynaložených pracovních dní, cestovní náklady nebo počítačové zdroje.
- *Počet výskytů konkrétní události* – příkladem událostí, které mohou být monitorovány jsou počet závad/chyb zjištěných během kontroly kódu, počet požadovaných změn požadavků anebo průměrný počet řádků kódu upravených v návaznosti na změnu požadavku.

První dva typy měření mohou být využity ke zjištění, zda procesní změny zapříčinily zvýšenou efektivitu provádění procesu. Příklad lze definovat následovně – mějme softwarový proces rozdělený do několika fází, tyto fáze jsou akceptace požadavků, dokončení návrhu architektury a dokončení vytváření testů a testovacích dat. Díky metrik je možné definovat čas a úsilí nutné k přesunu z jedné fáze do druhé a tak při úpravě procesu lze zjistit, zda došlo k redukci potřebného času a úsilí.

Měření počtu výskytů událostí má větší vliv na softwarovou kvalitu. Například zvýšení počtu nalezených chyb v programu po změně procesu inspekce kódu bude mít pravděpodobně přímý vliv na zvýšení kvality programu.

## 3.4 Metriky a zajištění kvality v rámci Automotive SPICE

Kvalita vývoje je při naplňování standardu Automotive SPICE velmi důležitá, z tohoto důvodu standard specifikuje k tomuto účelu množství procesů. Jedním z takových procesů je proces zajištění kvality SUP.1, jehož účelem je ujištění, že prováděné procesy a vzniklé pracovní produkty dodržují vytyčené vize a plány, a že nalezené neshody jsou řešeny a již nebudou v budoucnu znovu vznikat. Důležitým výstupem tohoto procesu jsou právě identifikované a zaznamenané shody a neshody s plány a vytyčenými cíli, které jsou poté prezentovány odpovědným lidem a ze kterých jsou dále vyvozeny důsledky a řešení. Mezi základní praktiky procesu SUP.1 patří vytvoření strategie zajištění kvality, což představuje vytvoření plánu, jak zjišťovat kvalitu. Další z několika praktik je zajištění kvality pracovních produktů a prováděných procesních aktivit [9]. Jedním ze způsobů, jak ověřovat kvalitu, jak pracovních produktů, tak i prováděných procesních aktivit, může být měření, které je specifikováno v dalším procesu Automotive SPICE.

Automotive SPICE specifikuje proces s názvem Měření a zkratkou MAN.6, jehož účelem je sběr a analýza dat vztahujících se k vyvíjenému produktu a implementovaným procesům v rámci organizace a jejích projektů. Tento proces podporuje efektivní správu procesů a demonstruje kvalitu produktu. Výsledkem takovéhoto procesu je to, že jsou identifikovány potřebné informace, které je nutné získat, a to pomocí v návaznosti na tuto identifikaci definovaných měření. Měření jsou poté prováděny a jejich výsledky sbírány, ukládány, analyzovány a poté vhodně interpretovány. Výsledné hodnoty metrik lze například využít pro podporu rozhodování v důležitých otázkách týkajících se projektu, pro které je daná metrika relevantní. Díky výsledků lze také identifikovat prostor pro zlepšení provádění daného procesu [10].

Mezi měření, prováděné v rámci procesu MAN.6, patří mimo jiné i metriky, které napomáhají sledovat a objektivně měřit kvalitu provádění různých procesů v rámci vývojového cyklu.

Potřeba konkrétních měření – metrik, vyplývá z jednotlivých procesů a jejich základních praktik, které je nutné dodržovat, a také i výstupů, kterých je nutné, v ideálním případě a při korektně provedeném procesu, dosáhnout. Metriky pomáhají hodnotit průběh procesu, stejně tak i konečné, dosažené výstupy procesu, a to vše i z pohledu historického, kdy je možné sledovat vývoj hodnoty metriky – vývoj kvality prováděného procesu.

Metriky, potřebné měřit a sledovat pro splnění standardu Automotive SPICE, se kterými se pracuje v rámci této práce, se týkají převážně domény požadavků a jiných úzce navazujících domén. Konkrétně se vybrané metriky týkají procesů SYS.1 až SYS.5, SWE.1, SWE.2, SWE.5 a SWE.6 definovaných v rámci Automotive SPICE a krátce popsaných v kapitole 2.4.1. Použité metriky lze rozdělit do dvou kategorií/typů. První kategorií jsou metriky, které sledují pokrytí (anglicky Coverage). Jedná se nejčastěji o poměr dvou čísel, jejímž výsledkem je reálné číslo, jinými slovy procentuální pokrytí. U tohoto druhu metrik je záhodno sledovat aktuální i historický stav hodnot a tím monitorovat průběh prováděného procesu. Příkladem takovéto metriky je pokrytí zákaznických požadavků systémovými požadavky, což znamená, že tato metrika vyjadřuje procento zákaznických požadavků, které jsou navázány alespoň na jeden systémový požadavek, což se dá také říct – kolik systémových požadavků vzniklo na základě zákaznických a existuje tedy mezi nimi logické propojení. Tato metrika se tedy počítá, jako poměr zákaznických požadavků ku systémovým požadavkům, vyjádřeno v hodnotě atributu – kolik zákaznických požadavků z celkového počtu má vyplněnou vlastnost „Satisfied by“. Metrika z kategorie pokrytí může také obsahovat více poměrů/pokrytí najednou, to znamená, že se v rámci jedné metriky počítá větší množství hodnot. Příkladem může být poměr požadavků s určitým vyplněným atributem, kdy každý jednotlivý poměr se týká jiného atributu, vyjádřeno na reálném příkladě – počet systémových požadavků s vyplněným atributem domény v poměru s celkovým počtem všech požadavků. Dalšími sledovanými atributy mohou být fáze, bezpečnost a další.

Druhou kategorií metrik, využitých v rámci této práce, jsou metriky množství (anglicky Number). Takovéto metriky sledují rozdělení entit (například požadavků) podle hodnoty jejího atributu, kdy vzniká množina čísel, vyjadřujících počet entit, jejichž atribut nabývá dané hodnoty. Příkladem použití může být metrika týkající se softwarových požadavků, kdy jsou požadavky rozděleny do skupin podle toho, v jakém stavu jsou – jestli má požadavek stav „ve vývoji“, „připravený na revizi“, „zrevidovaný“, „implementovaný“, „otestovaný“ nebo žádný stav nemá. Vhodným grafickým prvkem pro vizualizaci této metriky je sloupcový graf, čímž lze získat představu o aktuálním stavu sledované části projektu.

Pokud jsou získány výsledky měření, potažmo metrik, je záhodno verifikovat tyto výsledky. K tomuto účelu slouží Proces verifikace SUP.2, jehož účelem je potvrdit, že všechny pracovní produkty procesů splňují dříve specifikované požadavky. Výsledkem tohoto procesu je stanovená verifikační strategie, obsahující kritéria hodnocení z dané domény pracovních produktů, dále provedené verifikační aktivity a na to navazující identifikované nedostatky, které jsou dále zaznamenány a prezentovány zúčastněným stranám, včetně zákazníka [9].

# 4 Platforma IBM Jazz

IBM je jednou z největších technologických společností na světě, co se týká jak počtu zaměstnanců, tak i významnosti. Z výčtu jejího zaměření je vhodné zmínit vývoj hardwaru a softwaru, poskytování moderace a poradenských služeb a v neposlední řadě také výzkum, díky němuž firma IBM drží velké množství patentů. Mezi vynálezy IBM patří bankomaty, diskety, kreditní karty, relační databáze, programovací jazyk SQL a mnoho dalších [11]. Dnes patří mezi hlavní produkty a služby nabízené firmou IBM analytika, umělá inteligence, automatizace, cloudové výpočty, poskytování IT infrastruktury, bezpečnostní technologie a infrastruktura a v neposlední řadě také vývoj software [12]. Jedním z mnoha softwarových produktů firmy IBM je platforma Jazz, jejíž počátek vývoje se datuje až do roku 2005.

Jazz je platforma společnosti IBM pro zlepšení spolupráce mezi účastníky během celého životního cyklu softwaru nebo systému. Platforma IBM Jazz napomáhá k tomu, aby se dodání systému a softwaru stalo více produktivním, transparentním a kolaborativním procesem. Toho dosahuje integrací informací a úkolů napříč jednotlivými fázemi životního cyklu projektu [13]. Architektura platformy je navržen tak, aby netvořila překážky mezi různými nástroji, ale naopak je sdružovala pod jednou střechou. Oproti dřívějším monolitickým produktům/nástrojům přináší IBM Jazz inovativní přístup založený na otevřených a flexibilních službách a internetové architektuře.

IBM Jazz je webová platforma, vyžadující pro svůj běh server. Tento server je buď provozován společností IBM nebo lze využít vlastního serverového řešení a provozovat tedy platformu na svém serveru. Instalace aplikace je složitější a neobejde se bez využití mnoha manuálů. IBM Jazz je komerční platforma, tudíž pro její provozování je nutné zakoupit licenci.

Přístup k nástrojům platformy Jazz je možný pomocí webové aplikace nebo pomocí klientských aplikací založených na technologiích Eclipse a Microsoft Visual Studio. Dále existuje aspekt architektury platformy Jazz jménem „Linked Lifecycle Data“, který naplňuje standard Linked Data vytvořený konsorciem W3C. Standard Linked Data popisuje sémantiku a přistup v oblasti jednotného a standardizovaného propojení mezi systémy například pomocí formátu RDF [14]. Platforma naplňuje tento standard skrze přístup k datům z rozličných nástrojů pomocí specifikace OSLC. OSLC vytváří pojetí uživatelů/nástrojů jako poskytovatel a spotřebitel. Poskytovatel služby OSLC je nástroj, který vlastní data a poskytuje tato data jiným nástrojům pomocí RESTful rozhraní, jak je blíže popsáno ve specifikaci OSLC. Spotřebitel je nástroj, který přistupuje k datům jiného nástroje, pomocí rozhraní specifikovaného OSLC. Tímto je zajištěna interakce mezi nástroji různých dodavatelů i mimo platformu Jazz.

Další z mnoha možností přístupu je IBM Rational Lifecycle Integration Adapters – Tasktop Edition, jedná se tedy o adaptér, který umí integrovat nepřeberné množství nástrojů třetích stran, určených ke správě životního cyklu projektu. Tímto pomáhá týmům uchovat jejich stávající procesní infrastrukturu, pracovní postupy a ušetřit dříve vynaložené investice i v případě, že je nutné využít služeb jiného nástroje, v tomto případě platformy Jazz [15].

## 4.1 Nástroje platformy IBM Jazz

Platforma Jazz je složená z několika nástrojů, které jsou na sobě zcela nezávislé a to jak z pohledu logického napojení, tak i z pohledu datového, jelikož každý nástroj používá svou vlastní databázi. I přes oddělení jednotlivých nástrojů, nabízí platforma služby, které usnadňují hromadnou správu nástrojů a jimi poskytovaných služeb, jednou z nich je například služba pro správu uživatelů a jejich ověřování (autentizaci) při přihlášení a provádění úkonů.

Nyní následuje výčet nástrojů IBM Rational, které jsou od počátku vyvíjeny v rámci platformy Jazz [16]. Všechny tyto nástroje jsou nabízeny v jednom balíčku pod názvem IBM Collaborative Lifecycle Management (CLM) a jak už název napovídá, jedná se o sadu nástrojů vhodnou pro kompletní správu vývojového cyklu, které pomáhají týmu zlepšovat spolupráci, urychlovat dodání software a zlepšovat kvalitu software.

- *IBM Rational Team Concert* – pracovní prostředí pro spolupráci mezi vývojáři, softwarovými architekty a projektovými manažery se sjednocenými pracovními položkami, řízením zdrojového kódu a vydání verze a plánováním procesů a iterací.
- *IBM Rational DOORS Next Generation* – nástroj pro správu požadavků, který pomáhá týmu efektivněji definovat požadavky a účinně je využít v průběhu životního cyklu projektu pro získání lepších obchodních výsledků.
- *IBM Rational Quality Manager* – prostředí pro správu testů určené pro inženýry kvality, které nabízí nastavitelné řešení pro plánování testů, kontrolu pracovních postupů (workflow), sledování a reportování.
- *IBM Rational Rhapsody Model Manager a Design Management* – dva nástroje pro správu návrhu a architektury systému, které integrují návrh do celého aplikačního a systémového životního cyklu a umožňují tak týmům spolupracovat na návrhu.
- *IBM Rational Engineering Lifecycle Manager* – nabízí možnost vizualizovat, analyzovat a získat přehled o datech z životního cyklu projektu, pocházejících z různých nástrojů. Tým díky tomu lépe porozumí vztahům mezi daty a vytváří tak lepší a efektivnější rozhodnutí.

Mimo výše zmíněné nástroje existuje množství nástrojů IBM Rational, které mohou být jak klientským, tak i serverovým rozšířením výše zmíněných nástrojů. Jedná se například o Rational AppScan, Rational Performance Tester, Rational Software Analyzer, Rational Software Architect a mnoho dalších. Mimo nástroje IBM Rational dále existují rozšíření vzniklé v rámci programu IBM Rational Business Partners, kde partnerské firmy IBM poskytují integraci a rozšíření platformy Jazz.

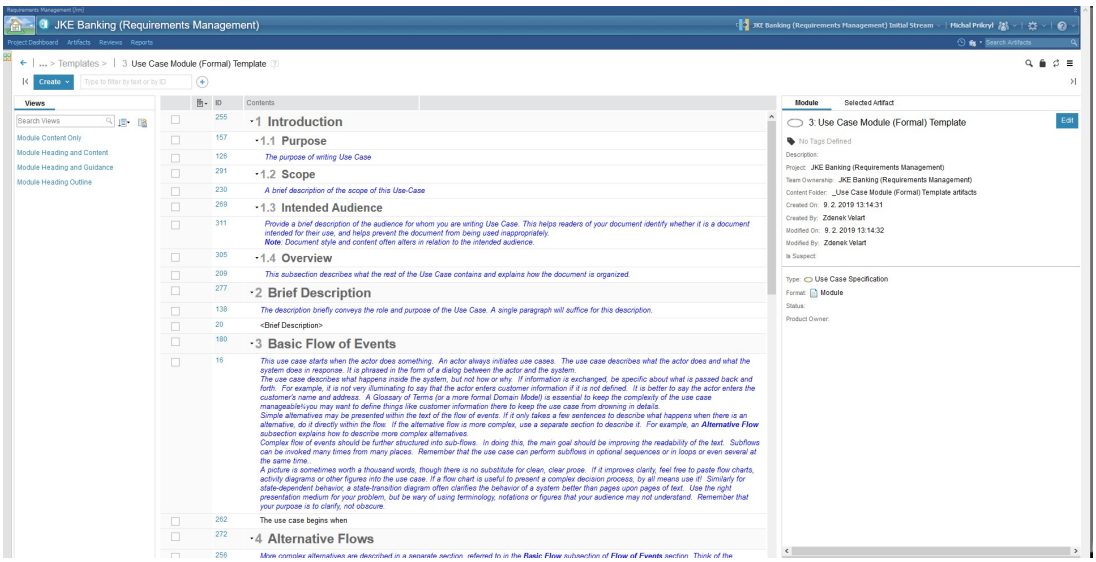
V rámci této práce, bude nejvíce využito nástroje IBM Rational DOORS Next Generation, jelikož ten je primárně určen pro správu požadavků a úkony spojené s nimi.

## 4.2 Nástroj IBM Rational DOORS Next Generation

IBM Rational DOORS Next Generation je nástroj pro správu požadavků, který nabízí lepší způsob jak definovat, analyzovat, sledovat a spravovat požadavky. Použití Rational DOORS Next Generation sebou přináší optimalizaci komunikace a spolupráce v týmu, což umožní zvýšit výslednou kvalitu a pracovat efektivněji [17].

Pro správu požadavků, speciálně jejich vytváření a editaci, nabízí IBM Rational DOORS Next Generation (dále IBM DNG) profesionální prostředí podobné tomu, jaké lze očekávat v textovém editoru typu Microsoft Word, a to i s výhodou sdíleného prostředí webové aplikace s výhodou již dříve zmíněné online spolupráce. Jednotlivé informace získané od zákazníka, které dále tvoří samotný požadavek – artefakt, lze uspořádat do modulů, čímž lze vytvořit logické skupiny mezi požadavky pro lepší vysledovatelnost a přehlednost. Artefakt lze také libovolně definovat/upravit na uživatelské úrovni – uživatel si může vytvořit svůj typ artefaktu, který lépe odpovídá jeho požadavkům na zásadní vlastnosti. Samozřejmostí je také možnost definice vlastních atributů artefaktu.

Na obrázku 5 lze vidět detail artefaktu, v tomto případě detail případu užití. V detailu je možné artefakt editovat, komentovat, zobrazit si návaznosti na artefakt a veškeré další informace spojené s daným artefaktem.

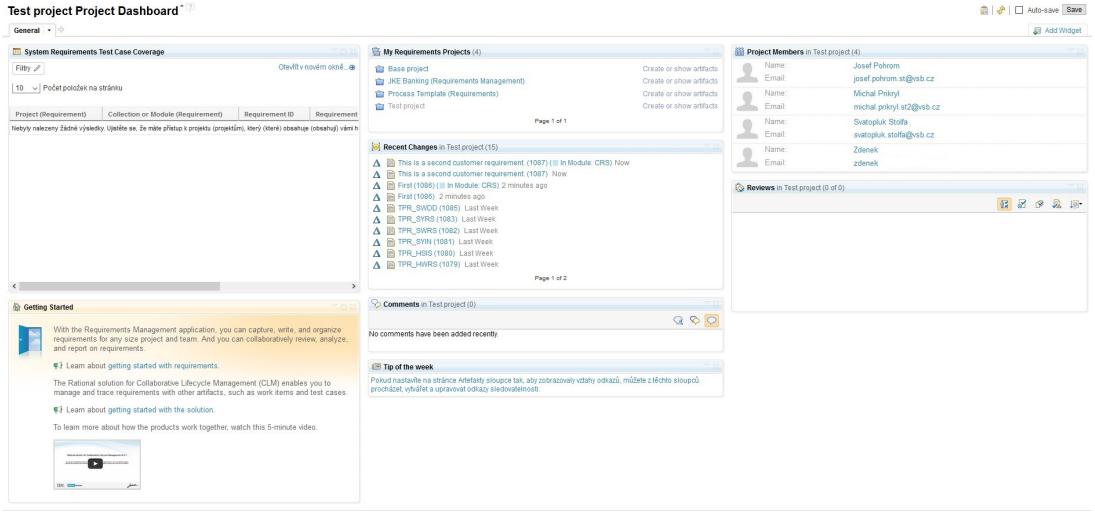



Obrázek 5: Detail artefaktu v nástroji IBM DOORS Next Generation

S rostoucí velikostí projektu se rozrůstá i počet požadavků a dalších artefaktů, s tím také vyvstane nutnost navázat požadavky na sebe a to ve vztahu, kdy jeden požadavek zpracovává, naplňuje nebo ověřuje jiný požadavek. Proto lze každý požadavek v IBM DNG navázat na jiný artefakt v rámci projektu, jako například jiný požadavek, testovací případ atp. Pochopení takovýchto vazeb je důležité ke správnému porozumění projektu, z tohoto důvodu nástroj nabízí stromový náhled na návaznosti mezi artefakty. Dalším pomocníkem pro porozumění návazností v

projektu je grafické zobrazení, kde je možné vidět návaznosti na daný artefakt ve formě podobné grafu/sítí s vrcholy a hranami.

Jednou z dalších výhod je možnost upravit si nástěnku projektu dle vlastních požadavků. Nástěnka, viz obrázek 6, nabízí přehledný prostor pro zobrazování rychlého přehledu o vývoji, trendových metrikách zobrazených v grafu a dalších vlastností nebo stavů projektu. Další možností je zobrazení změn projektu, kde je možné zobrazit změny čekající na potvrzení nebo změny již potvrzené jiným administrátorem. Na tuto agendu se váže vlastnost nástroje IBM DNG, kdy každá změna artefaktu je sledována a zaznamenána, a to z toho důvodu, aby mohla být někým dalším (například systémovým analytikem) schválena, aby se předešlo nevhodným změnám a úpravám, které mohou způsobit problémy při následném vývoji a práci na projektu. V případě nedostatečné či chybějící funkcionality nabízí nástroj možnost vytvořit si své vlastní rozšíření/doplňkové moduly v jazycích HTML a JavaScript, které lze poté importovat a zobrazovat na nástěnce a tím získat požadované specifické informace nebo provést nadstavbové operace a úpravy s daty.




Obrázek 6: Nástěnka projektu v nástroji IBM DOORS Next Generation

Jak již bylo zmíněno, nástroj uchovává informace o všech změnách provedených na každém jednotlivém artefaktu, a to kdo změnu provedl, kdy ji provedl a co změnil. Díky tomu lze získat report/výstup o práci na projektu, frekvenci změn v projektu anebo pouze informace, kdo provádí jaké úkony a zda jsou v jeho kompetenci. Díky této funkcionalitě je také možné zachytit snímek (snapshot) projektu, a tak uchovat záznam o stavu projektu při vydání určité verze vyvíjeného produktu a sledovat například provedené změny v produktu mezi minulou a následující vydanou verzí.

Nástroj IBM DNG také nabízí možnost vizualizovat obchodní procesy pomocí digramu obchodních procesů. Tyto diagramy a další diagramy jako například diagram případu užití a další, lze vytvářet přímo v nástroji. Tvorba a zkoumání takovýchto diagramů napomáhá k lepšímu porozumění daného problému.

Z oblasti měření kvality software a metrik nenabízí nástroj IBM DNG moc funkcionality. Vizualizace je v omezené míře možná pouze na nástěnce, kde je možné využít widgetu, který je ale nutné si vytvořit nebo získat z externího zdroje. Takový widget může zpracovávat zadané data a z nich poté vytvářet grafy, problémem ale zůstává, že je nutné tento widget vytvořit, není tedy základní součástí nástroje. Jak již tedy vyplývá, nástroj neumožňuje definovat žádné pokročilé uživatelské metriky.

Jak již bylo zmíněno, pro rozšíření nástroje lze využít OSLC. Platforma Jazz dále nabízí nástroj s názvem Tvůrce sestav, ve kterém je možné vytvořit sestavu z požadovaných dat, která je poté přístupná ke stažení pomocí jedinečné URL adresy nebo přes widget na nástěnce projektu. Tuto sestavu lze využít ke generování dat, ze kterých se poté počítá nebo sestavuje metrika. Bohužel základní widget nabízí pouze tabulkové zobrazení hodnot, nikoli grafické a také zde chybí možnost historického přehledu změn metriky. Nyní tedy nastává prostor pro rozšíření, které je mimo jiné předmětem této práce. Toto rozšíření se bude napojovat na nástroj IBM DNG a buď pomocí OSLC nebo pomocí sestav získá data potřebné pro sestavení metriky, které poté v oddělené webové aplikaci zobrazí v grafické podobě uživateli, čímž vyřeší chybějící funkcionalitu IBM DNG v oblasti měření kvality software a metrik.

# 5 Srovnání existujících nástrojů a aplikací

V této kapitole následují analýzy několika nástrojů pro práci s požadavky, které umožňují rozšíření platformy IBM Jazz v oblasti metrik a měření kvality. Nástroje nejsou přímými rozšířeními platformy IBM Jazz, ale umožňují datové napojení. Přímé rozšíření platformy IBM Jazz, potažmo nástroje IBM DOORS Next Generation, jsou ve valné většině případů neveřejná a každá společnost je používá pouze pro své vlastní potřeby a neumožňuje využití komunitě uživatelů a dalších společností. Nalezen byl pouze jeden nástroj – Metrics Dashboard od společnosti River North Solutions INC. [18], který se jevil jako využitelné rozšíření, bohužel ale nebylo možné jej nijak získat, protože tento nástroj není dostupný ke stažení. Dle popisu se jedná o nástroj generující metriky v daných časových intervalech, který bohužel disponuje pouze pevnou množinou metrik, které jsou předdefinovány a možnost přidání uživatelských metrik není možná. Další z nevýhod je to, že se jedná o lokální aplikaci, která je spustitelná pouze ve formě Windows servisu a reporty (výsledky metrik) generuje pouze ve formátu HTML souboru, což také nesplňuje stanovené požadavky na jednoduchý přístup k výsledným informacím. Nástroj Metric Dashboard tedy nelze rozšířit, ať už z pohledu nedostupnosti implementace, tak i z pohledu nevhodné architektury nástroje.

Existuje velké množství nástrojů, které umožňují správu požadavků v rámci projektu. Mezi těmito nástroji lze najít nástroje více či méně pokročilé, minimálně, co se týče možností práce s požadavky a možností definice a zobrazení metrik. Nespornou výhodou takovéhoto nástroje je to, zda umí pokrýt kompletně celý vývojový proces a všechny jeho fáze, tudíž nebude nutné používat více nástrojů od různých tvůrců. Další nespornou výhodou by také mělo být, že nástroj bude podporovat online kolaboraci. To znamená, že projekt, a v lepším případě i nástroj, bude dostupný online, popřípadě přes webovou aplikaci. Hlavním rozdílem je také pořizovací cena nástroje. Většina zkoumaných nástrojů je komerčních, čili placených, proto bylo tedy ve většině případů využito krátkodobých zkušebních verzí.

Následující vybrané nástroje byly dle autorova názoru těmi lepšími z množiny všech existujících nástrojů pro správu požadavků, a proto byly také pro srovnání vybrány. Srovnání každého nástroje se skládá ze čtyř částí. První částí je komplexní popis nástroje, druhou částí je analýza a seznámení s nástrojem, ve třetí části jsou prozkoumány možnosti správy požadavků v nástroji spolu s průzkumem agendy metrik a nakonec v poslední části, se nachází celkové zhodnocení nástroje.

## 5.1 codeBeamer ALM

Nástroj codeBeamer ALM nabízí sadu nástrojů pro kompletní správu životního cyklu aplikace od počátku projektu, až po vydání funkční verze. Součástí nástroje codeBeamer ALM jsou správa požadavků, nástroj pro podporu vývoje software, správa testů a kvality, risk management, správa DevOps a další. Nástroj je zaměřen převážně na agilní metody vývoje, ale je možné jej využít i při jiných metodikách (například vodopádové aj.) [19].

### 5.1.1 Analýza nástroje codeBeamer ALM

Architektura codeBeamerALM je založena na tom, že jednotlivé nástroje nejsou rozděleny do menších částí, ale všechny jsou součástí jedné aplikace – platformy, která se tímto stává jednotným místem pro správu projektu při všech fázích vývoje, podobně jako je tomu u IBM Jazz. Samozřejmostí je, že nástroj lze rozdělit na jednotlivé části a používat pouze tu, kterou potřebujeme a tím neplatit další nepotřebné nástroje. Integrace všech modulů v jedné aplikaci pomáhá zajistit transparentnost, viditelnost a vysledovatelnost procesů dokonce i v tom nejsložitějším projektu. CodeBeamer ALM nabízí také předdefinované šablony projektů pro zdravotnictví, automobilový průmysl (mimo jiné i Automotive SPICE), letecký průmysl a farmaceutický průmysl. Tyto šablony podporují dodržování bezpečnostních i jiných standardů, mimo jiné také pomáhají s dosažením procesní vyspělosti, dodržováním předpisů, zkracují dobu vývoje a tím pomáhají snížit celkovou cenu vývoje.

Nástroj codeBeamer ALM lze provozovat jako serverovou aplikaci, hostovanou buď na vlastním serveru nebo na serveru vydavatele softwaru, v obou případech přístupnou online přes webový prohlížeč. Jelikož se jedná o komerční nástroj, bylo využito třicetidenní online zkušební verze, kde je možné si vyzkoušet plně funkční aplikaci codeBeamer ALM bez nutnosti pořízení licence.

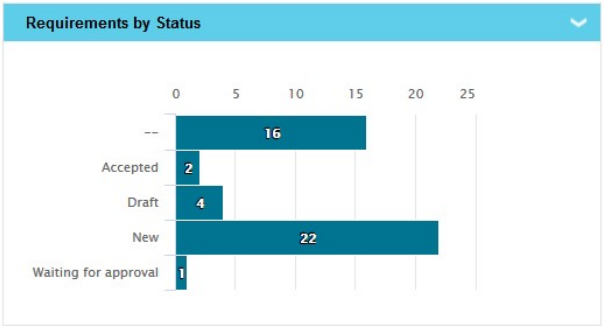
Častým problémem je, že vývojový tým a zákazníci využívají různé nástroje pro stejný účel nebo například nástroje, nabízející funkce, které v codeBeamer ALM nejsou dostupné. Z tohoto důvodu nabízí codeBeamer ALM integraci s mnoha různými aplikacemi a platformami jakými jsou například IBM Rational DOORS, Microsoft Word a Excel, Slack, Docker, Enterprise Architect, JIRA, JUnit, Selenium, Git a další, zkrátka nástroje používané při všech fázích vývoje softwaru.

### 5.1.2 Správa požadavků v nástroji codeBeamer ALM

CodeBeamer ALM nabízí pokročilou správu požadavků, která zajišťuje vysledovatelnost návazností požadavků, spolupráci při vytváření požadavků a možnost vytvářet a sledovat pokrytí kódu a testovacích případů požadavky. Požadavky mohou být propojeny se všemi artefakty vznikajícími napříč celým životním cyklem projektu (například zdrojový kód, úkoly, chyby, testy atp.). Je možné zobrazit návaznosti požadavku v rámci celého vývojového cyklu, pomocí k tomuto účelu vhodných grafických a textových struktur. Nástroj nabízí také import, export, sdílení a znovupoužití požadavků s dalšími účastníky projektu, kteří nemají přístup do tohoto nástroje. Samozřejmostí je také napojení jiného nástroje, jak již bylo zmíněno v předchozí podkapitole, pomocí flexibilního REST API.

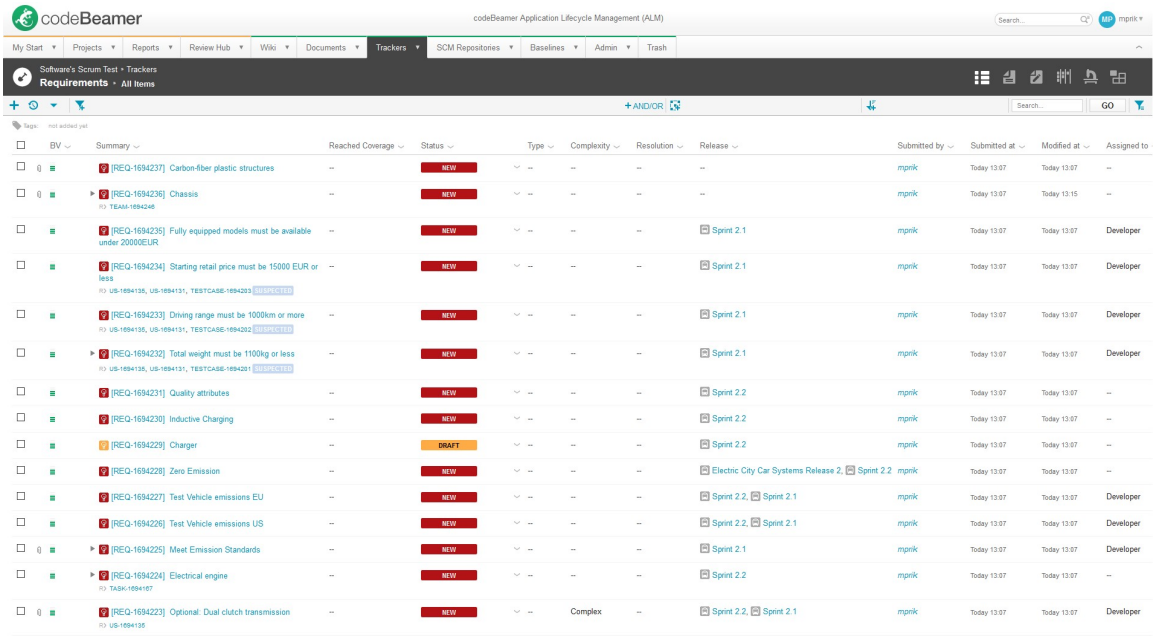
Správa požadavků se nachází pod záložkou menu „Trackers“, kde je možné pracovat se všemi pracovními položkami, mezi které požadavky patří. Po kliknutí na záložku je zobrazena nastavitelná nástěnka s grafickými přehledy o pracovních položkách a jejich provázání, plnění,

stavech atp. Na obrázku 7 je možno vidět rozdělení požadavků dle jejich stavu, zobrazené v grafické formě na nástěnce.




Obrázek 7: Grafické znázornění rozdělení požadavků dle stavu z nástěnky codeBeamer ALM

Při rozvinutí seznamu požadavků se přehledně vypíší všechny dostupné požadavky přiřazené k aktuálně prohlíženému projektu. Je možné měnit zobrazení seznamu požadavků a to do formy tabulky (ukázka na obrázku 8), dokumentu, dokumentu s možností editace, nástěnky (Kanbanu), pokrytí požadavků testovacími případy a vysledovatelnosti v návaznosti na další artefakty.




Obrázek 8: Uživatelské rozhraní webové aplikace codeBeamer ALM pro správu požadavků

Tabulka všech požadavků je vhodná pro komplexní přehled o stavu požadavků a jejich atributech (viz obrázek 8). Zobrazení požadavků ve formě dokumentu, je více detailní a přehlednější než tabulkové, umožňuje přidání komentáře k požadavku a přiřazení požadavku k vydané verzi nebo k testovacímu případu. Zobrazení formou dokumentu s možností editace je sloučením dvou předchozích s tím rozdílem, že je možné editovat veškeré atributy požadavku. Zobrazení formou

Kanbanu je určené pro správu požadavků stylem drag and drop, kdy je možné přetažením měnit stav požadavku.

Záložka pokrytí testy znázorňuje, kolik požadavků je pokryto testovacími případy (zkontrolováno jejich zapracování v softwaru) a ve formě tabulky přehledně zobrazí jednotlivé požadavky a informace o jejich otestování.

V záložce prohlížeč vysledovatelnosti lze zobrazit provázání jednotlivých požadavků (možno i jiných artefaktů a pracovních položek) s jinými druhy artefaktů a pracovních položek. Do jednotlivých úrovní je možné nastavit požadovaný druh artefaktu ke znázornění, kdy je poté v tabulce zobrazeno propojení jednotlivých artefaktů z vybraných úrovní. Tímto způsobem lze například zobrazit propojení mezi požadavky, testovacími případy a chybami.

Samotné požadavky je možné definovat pomocí množství atributů. Přidanou hodnotou k množství atributů je také dříve zmíněná možnost propojení mezi artefakty, čímž je možné požadavky ještě lépe definovat. Požadavky je také možné propojovat navzájem mezi sebou. Při zobrazení požadavků ve formě dokumentu lze všechny atributy vidět v pravé části stránky, kde se nachází i ostatní informace o požadavku, jako komentáře a propojení na jiné artefakty.

Z oblasti měření kvality a metrik lze najít v nástroji codeBeamer ALM výpisy aktuálního stavu projektu – metrik, ale většinou, kromě grafického znázornění na nástěnce, pouze v textové podobě, jako například pokrytí požadavků testovacími případy atd. Další možností je záložka týkající se vysledovatelnosti, kde lze znovu vidět pouze textové vyhodnocení, nikoliv grafické. Také časový průběh metriky od minulosti po současnost není v nástroji dostupný, tudíž nelze sledovat vývoj, pokud si jej uživatel nebude ukládat a poté ručně porovnávat, což se jeví neprakticky.

Pod záložkou hlavního menu „Reports“, lze pomocí dotazovacího jazyka cbQL, velmi podobného jazyku SQL, filtrovat vybrané artefakty, seskupovat a řadit dle vybraných atributů a tak získat další informace o stavu projektu. Výhodou je také to, že report (dotaz v cbQL), lze uložit pro budoucí použití, tudíž není nutné dotaz pokaždé vytvářet, ale pouze si jej načíst a zobrazit si jeho výsledek, čímž lze také získat data pro metriky, ale znovu je nutné je pokaždé ručně sestavit a vypočíst.

### 5.1.3 Zhodnocení nástroje codeBeamer ALM

Hlavní výhodou nástroje codeBeamer ALM a jeho správy požadavků je možnost zachycení všech fází vývoje a s nimi spojených artefaktů, čímž je možné získat provázanost mezi požadavky a dalšími pracovními produkty a artefakty, čehož se dále využívá při pokročilé správě požadavků. Nástroj se dále pyšní přehledným uživatelským rozhraním, ve kterém lze vše jednoduše najít. Mírnou nevýhodou je cena, ale ta je v tomto případě adekvátní kvalitě. Další mírnou nevýhodou jinak výborného nástroje je také to, že zde neexistují pokročilejší možnosti definice a vizualizace metrik. Vizualizace je možná pouze textová, navíc v omezené míře, kdy není možné definovat vlastní metriky. Tato funkčnost by do jisté míry mohla být dostačující, ale ne pro požadovaný účel a funkčnost.

## 5.2 Enterprise Architect

Enterprise Architect od společnosti Sparx Systems je kompletní nástroj pro systémovou analýzu a návrh, který pokrývá všechny fáze vývoje projektu od počátku, tj. sběru požadavků, až po testování a údržbu, to vše s pomocí přehledných UML diagramů a jiných způsobů znázornění. Hlavním účelem nástroje je návrh a konstrukce softwarového systému, modelování business procesů a modelování v různých průmyslových doménách. Díky rozsáhlému zaměření je nástroj určen širokému spektru uživatelů od programátorů a business analytiků až po enterprise architekty [20].

### 5.2.1 Analýza nástroje Enterprise Architect

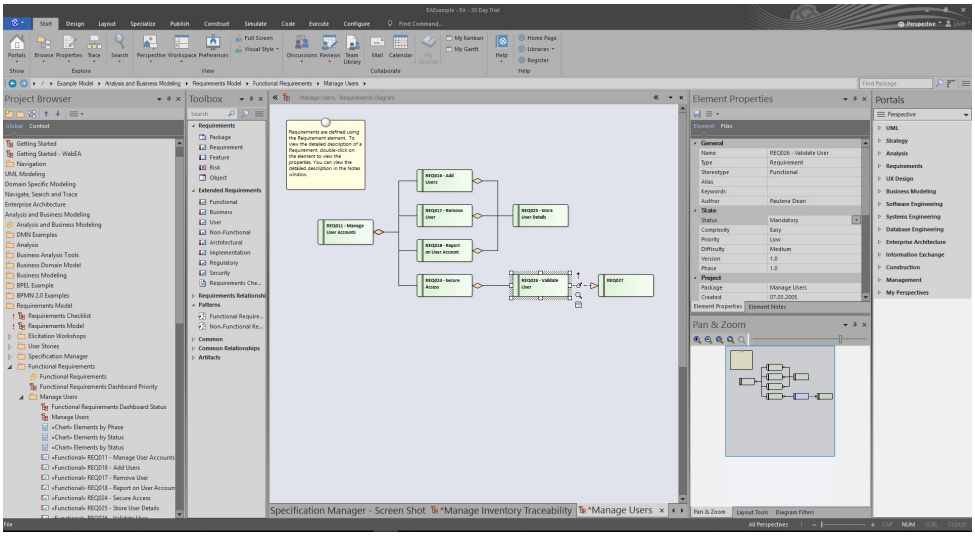
Nástroj Enterprise architect je určen jak pro podporu při návrhu a implementaci nového systému, tak i při změně systému již existujícího. Hlavními prvky nástroje jsou funkce pro správu požadavků a také další funkcionalita pro ostatní fáze vývoje – návrh, konstrukci, testování a údržbu. Výhodou je také podpora vysledovatelnosti, možnost správy projektu a změn. Samozřejmostí je také podpora množství průmyslových standardů pro modelování softwaru a business systémů, kterými jsou například UML, SysML, BPMN, WSDL, OWL a mnoho dalších. Dále také podpora standardů vývoje pro odvětví letectví, biomedicíny, telekomunikací, automotive a dalších. Výhodou Enterprise Architect je také to, že obsahuje integrované vývojové prostředí, které podporuje editaci kódu se zvýrazněním syntaxe a intellisense pro jazyky jako C, C++, C#, Java, PHP a další. V neposlední řadě také nabízí integraci s dalšími nástroji a službami a to pomocí importu/exportu CSV souborů nebo automatických rozhraní typu API nebo OSLC nebo přímé napojení na nástroje jako IBM DOORS, JIRA, TFS, bez nutnosti uživatelské implementace.

Distribuce nástroje Enterprise Architect probíhá pomocí spustitelného „.exe“ souboru, který je nutné spustit a poté nainstalovat na svou pracovní stanici. Enterprise Architect je komerčním programem, tudíž je na jeho stažení a provozování nutné vlastnit placenou licenci. Pro vyzkoušení je nabízena zkušební třicetidenní verze, kde je možné vyzkoušet kteroukoli z nabízených edicí (Professional, Corporate, Unified, Ultimate). Edice se liší množstvím dostupných funkcí a také počtem implementovaných změn a vylepšení. Pro potřeby analýzy bylo využito edice Ultimate.

### 5.2.2 Správa požadavků v nástroji Enterprise Architect

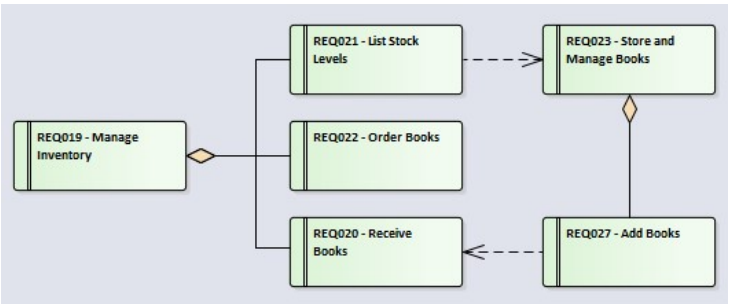
Uživatelské rozhraní nástroje Enterprise Architect (dále jen EA) je podobné rozhraním programů rodiny Microsoft Office, kdy hlavním rozdílem je vyšší složitost a členění uživatelského rozhraní EA oproti MS Office, jak je možno vidět na obrázku 9.

Mezi hlavní funkce správy požadavků v EA patří možnost přizpůsobení, jak budou požadavky dokumentovány, dále propojení požadavků k návrhovým a implementačním prvkům a také zajištění vysledovatelnosti požadavků ve fázích návrhu a implementace. Požadavky se mohou stát součástí řízení změn, pracovních postupů a auditů.




Obrázek 9: Uživatelské rozhraní nástroje Enterprise Architect

Po načtení ukázkového projektu lze ve stromové struktuře v levé části obrazovky, pod prvkem Analýza a business modelování, najít model požadavků, kde se nachází všechny dostupné požadavky. Požadavky je možné seskupovat do libovolných logických skupin (složek), tím je lze roztřídit například na funkční a nefunkční a dále třídit například dle případů užití. Každý požadavek obsahuje základní atributy jako krátký popis, status, prioritu, složitost a další. Hodnoty těchto požadavků jsou dále, mimo jiné, využívány při vyhodnocování kvalit vývoje. Uživatel má také možnost vytvořit si vlastní atributy požadavku. Díky pokročilé možnosti modelování je možné požadavky vizuálně skládat do podoby diagramů, kde jsou přehledně vidět návaznosti na jiné požadavky a další pracovní položky (například testovací případy nebo části zdrojového kódu). Příklad modelování vzájemné návaznosti mezi požadavky (s využitím agregace) lze vidět na obrázku 10. Samozřejmostí je zobrazení všech požadavků formou tabulky/dokumentu, kde lze přehledně vidět nejdůležitější atributy požadavku, kterými jsou název, popis, status aj.



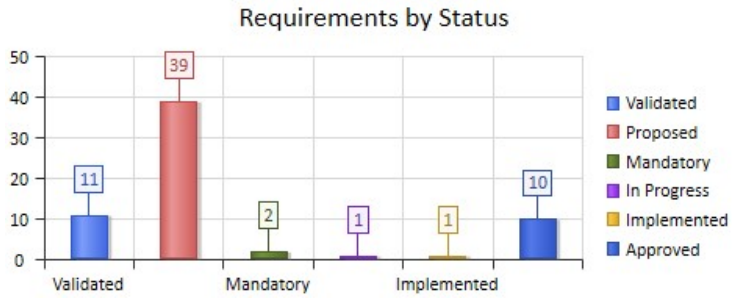

Obrázek 10: Příklad modelu návaznosti mezi požadavky v Enterprise Architect

Komunikace s účastníky vývoje je jednodušší s možností generování reportů ve formátu HTML, RTF nebo PDF. Import požadavků od zákazníků nebo z jiných systémů je také možný díky možnosti přidání knihoven třetích stran, které tyto funkce umožňují. Druhou možností pro

synchronizaci s účastníky nebo jinými vývojovými týmy je DOORS MDG Link. Tento nástroj poskytuje nenáročné propojení mezi EA a IBM DOORS, které umožňuje kombinovat požadavky vzniklé v IBM DOORS s požadavky vzniklými v EA a to pomocí vzájemného importu a exportu. Propojení sebou přináší benefit v podobě doplnění chybějících funkcí a možností jednoho nástroje funkcemi a možnostmi nástroje druhého. Dalšími benefity jsou možnost výběru atributů, které budou přeneseny a možnost synchronizace, při zachování použití rozdílných prostředí pro každý pracovní tým [21].

Analyzovat a sledovat požadavky, jinými slovy sledovat kvalitu a metriky vývoje, je možné díky oknu vysledovatelnosti, kde je možné ve vztahové matici vidět pokrytí (vztah) jednotlivých požadavků případy užití nebo jinými navazujícími artefakty.

Generování grafických výstupů s informacemi o požadavcích a jejich metrikách je také jednoduché. Na výběr je několik druhů grafů, ať už předdefinovaných (koláčový, sloupcový aj.), tak i dalších, které si uživatel může vytvořit. Pomocí grafů lze zobrazit rozdělení požadavků dle vybraného atributu, například kolik požadavků z celkového počtu je v jakém stavu, viz ukázka na obrázku 11. Tímto grafem lze například přehledně zobrazit podklad pro měření kvalitativní metriky plnění požadavků, která měří poměr počtu požadavků implementovaných k celkovému počtu požadavků v rámci projektu.




Obrázek 11: Graf znázorňující rozdělení požadavků dle jejich statusu

### 5.2.3 Zhodnocení nástroje Enterprise Architect

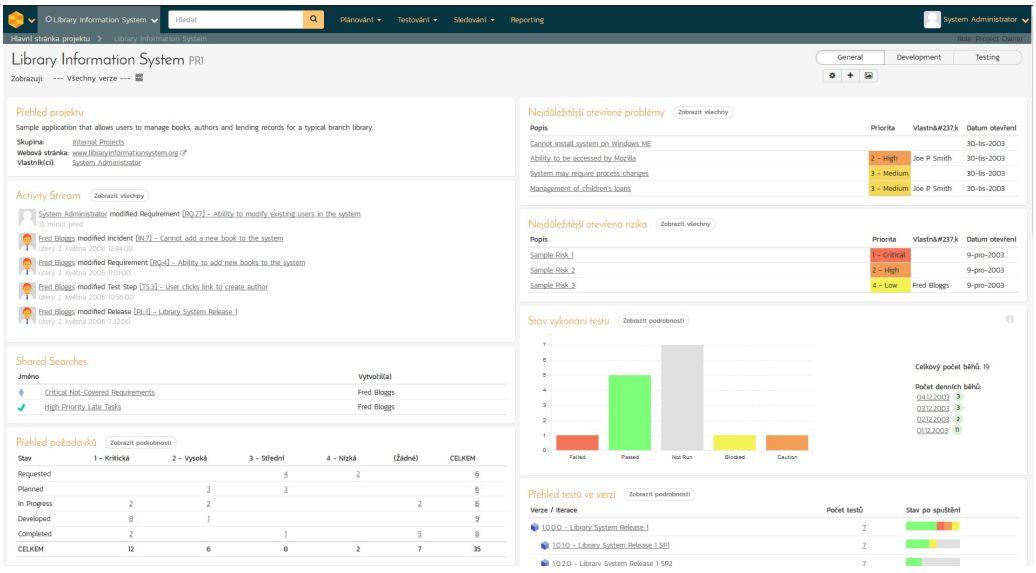
Enterprise Architect je rozsáhlý nástroj pro podporu softwarového procesu. Nabízí mnoho možností uživatelských úprav a celkově nechává uživatele volněji pracovat a upravit si prostředí nástroje přesně pro aktuálně používaný proces. Z hlediska správy požadavků nabízí EA širokou škálu možností vizualizace požadavků a jejich vztahů a návazností, které lze poté dále využít při vyhodnocování kvality softwarového vývoje a procesu obecně. Bohužel znovu neumožňuje definice uživatelských metrik a ani podrobnější agendu a zobrazení metrik v takové míře, aby byla dostačující požadovanému účelu rozšíření IBM Jazz.

## 5.3 SpiraTeam

Nástroj SpiraTeam od společnosti Inflectra Corporation je systém pro správu životního cyklu aplikace. SpiraTeam pokrývá celý životní cyklus, což zahrnuje vše od požadavků, úkolů, chyb, testů až po úkony doprovázející vydání verze aplikace a to vše v jednom integrovaném prostředí. Jednou z výhod je lokalizace prostředí nástroje do českého jazyka. Mezi hlavní funkce nástroje SpiraTeam patří pokročilé a velmi rozsáhlé možnosti vizualizace všech možných artefaktů projektu. Další hlavní funkcí je možnost generování reportů v různých formátech jako HTML, PDF, DOC, DOCX a XML [22].

### 5.3.1 Analýza nástroje SpiraTeam

Rychlý přehled o stavu projektu lze jednoduše získat na hlavní stránce projektu, kde je formou grafů a krátkých výpisů popsán aktuální stav jednotlivých organizačních částí projektu, ukázku hlavní stránky lze vidět na obrázku 12. Samozřejmostí je podpora agilních a jiných metod vývoje spojená s release managementem, dále je také umožněno členění vývoje na sprinty a iterace, jejímž výsledkem je vždy nová, spustitelná verze aplikace. Nástroj umožňuje také integraci zdrojového kódu, kdy je možné provádět revize kódu spolu s připojením kódu k dané revizi, čímž je možné vysledovat změny kódu spojené s uživatelskými a jinými požadavky, incidenty a dalšími artefakty vývoje. Komponenta s názvem plánovací nástěnka umožňuje správu požadavků ve formě podobné Kanbanu, kde je možné jednotlivé požadavky přehledně spravovat, třídit pomocí různých atributů, rozdělovat do skupin nebo zobrazit podrobnosti a k požadavku přiřazené úkoly.




Obrázek 12: Hlavní stránka projektu v nástroji SpiraTeam

Správa úkolů má také své místo v rámci nástroje. V této části lze zaměstnancům přiřazovat úkoly, dále je organizovat a sledovat jejich pokrok v reálném čase. Každému úkolu může

být přiřazen úkolem řešený požadavek, čímž je znovu podpořena vysledovatelnost požadavků v projektu. Další část – správa zdrojů, obsahuje informace o zaměstnancích, jejich pracovním zařazení, celkovém i rozepsaném vytížení na projektu a možnou budoucí alokaci pro další práci na projektu.

Velkou část nástroje tvoří funkce potřebné pro testování. Výhodou je znovu možnost propojení testů a testovacích případů s dalšími artefakty vývoje, jakými jsou požadavky a vydané verze. Každý testovací případ je možné podrobně krok po kroku popsat, připojit k němu testovací skripty nebo navázat automatické testy, dále také přiřadit výsledky a sledovat jejich vývoj. Nástroj podporuje integraci dalších nástrojů nutných pro automatické testování, jakými jsou například Selenium, QuickTest Pro a další. S testováním je také spojena agenda chyb, které vyvstaly v průběhu testování.

SpiraTeam nabízí systém pro online správu dokumentů, kde je možné centralizovaně nahrávat a upravovat dokumenty a tyto dokumenty připojovat k jiným artefaktům jako například požadavkům, incidentům a úkolům. Komunikace mezi členy projektu je možná pomocí integrovaného komunikačního kanálu, kde spolu uživatelé mohou komunikovat pomocí krátkých zpráv.

Další výhodou je také to, že SpiraTeam nabízí napojení na služby a nástroje typu IBM DOORS, Enterprise Architect aj. tak, jako ostatní konkurenti. SpiraTeam také nabízí velké množství dalších přídavných modulů pro napojení na většinu nejpoužívanějších služeb z oblasti testování, vývoje softwaru, systémů pro správu chyb a systémů pro správu zdrojového kódu. V dnešní době je již nutností podpora mobilních aplikací, tuto možnost SpiraTeam nabízí a to pro iOS, Android a Windows Phone.

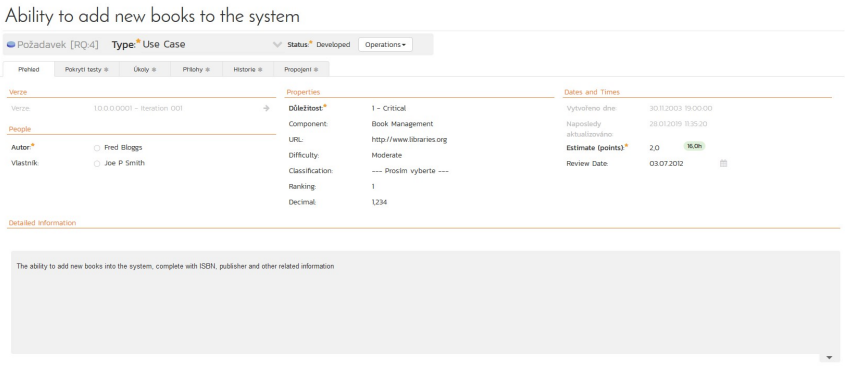
SpiraTeam je distribuována k uživatelům jako moderní webová aplikace, tudíž s výhodou, že není nutné cokoli instalovat na svou vlastní pracovní stanici a také s přístupem k práci na projektu odkudkoli. Jedná se o komerční nástroj, ale jako jiné podobné nástroje také nabízí třicetidenní lhůtu, kdy je možné využívat bezplatně plnou verzi nástroje.

### 5.3.2 Správa požadavků v nástroji SpiraTeam

SpiraTeam nabízí velmi pokročilé a propracované prostředí pro správu požadavků. Jako první stránka v sekci požadavků se nachází nástěnka, zde je možné zobrazit a filtrovat všechny požadavky přiřazené k projektu. Požadavky je možné seskupovat do balíčků, čímž je lze rozdělit do lépe spravovatelných logických celků. Jednotlivé požadavky jsou zobrazeny se všemi svými atributy a návaznostmi – testovacími případy, úkoly pro tým a verzemi aplikace, které implementují daný požadavek. Jednotlivé požadavky lze jednoduše editovat přímo na hlavní nástěnce.

Popis a atributy jednotlivých požadavků jsou velmi detailní a poskytují plně dostačující prostor pro zaznamenání informací o požadavku, jak je ostatně vidno na obrázku 13. Detail požadavku nabízí mimo možnosti rozšířené editace atributů také záložku s pokrytím testů, kde lze vidět všechny testy spojené s požadavkem, spolu s informacemi o testu a jeho posledním spuštění, kdy agendě testů se plně věnuje samostatná sekce nástroje SpiraTeam. Další záložkou v detailu požadavku jsou úkoly, kde se nachází úkoly, spolu s jejich postupem a stavem. Tyto úkoly

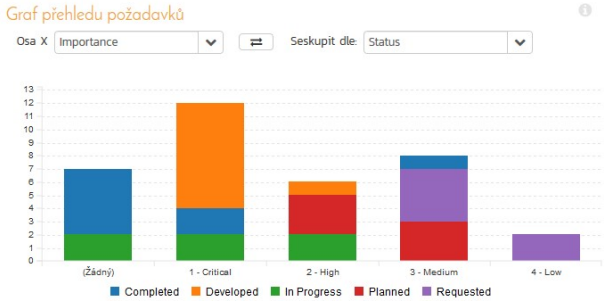
jsou přiřazeny k danému požadavku, tudíž jej plně nebo z části pokrývají a plní. Ke každému požadavku lze také přiložit dokumenty a jiné přílohy, které napomáhají popisu a pochopení daného požadavku. Nezbytnou částí nutnou pro podporu vysledovatelnosti požadavků v projektu je záložka historie, kde lze vidět všechny změny požadavku, kdo je provedl, co změnil a kdy. V poslední záložce jsou vypsána všechna propojení na aktuální požadavek i s komentářem a stavem. Propojení lze vytvořit na libovolný artefakt v rámci projektu i v rámci všech ostatních projektů, čímž je dosaženo možnosti znovupoužití artefaktů – požadavků mezi projekty.




Obrázek 13: Detail požadavku v nástroji SpiraTeam

V případě, že atributy požadavku či jeho popis nejsou dostatečně jasné, je možné transformovat požadavek na typ případ užití (use case) nebo scénář, čímž je přidána možnost definice jednoho nebo více kompletních případů užití i s posloupností kroků vedoucích k jeho naplnění. SpiraTeam poskytuje do jisté míry i možnost definovat vlastní typy a stavy požadavků, což napomáhá pokrytí požadavků i specifičtějšího a komplikovanějšího projektu.

Z oblasti měření a metrik požadavků nabízí nástroj SpiraTeam přehledné grafy, kde lze v reálném čase sledovat různé rozdělení požadavků na základě výběru uživatele. Lze nastavit hodnotu na ose X a seskupování požadavků ve sloupci podle atributu na ose X. Příklad na obrázku 14 znázorňuje požadavky rozdělené dle důležitosti a dále dle jejich statusu. Nevýhodou je to, že nástroj neumožňuje pokročilejší správu, zobrazování nebo definici metrik. Jedinou možnosti je pouze zobrazení grafů a manuální kontrola pozorovaných hodnot zobrazených v grafu.




Obrázek 14: Přehled požadavků ve formě grafu v nástroji SpiraTeam

### 5.3.3 Zhodnocení nástroje SpiraTeam

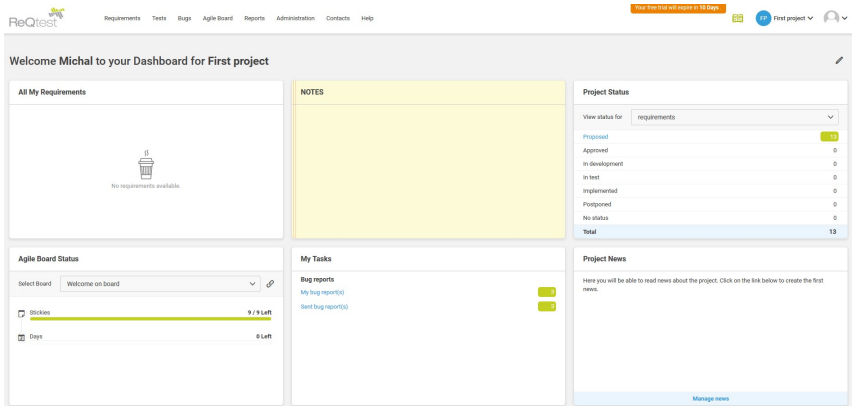
SpiraTeam je velmi kvalitní a uživatelsky přívětivý nástroj, který splňuje mnoho požadavků pro kvalitní, standardizovaný vývoj. Správa požadavků a zobrazování metrik v reálném čase je taktéž velmi dostačující, stejně tak i možnosti integrace artefaktů z jiných nástrojů a aplikací a tím oproštění od zdlouhavého kopírování nebo ztráty investice do jiného nástroje. Díky možnostem propojení artefaktů lze získat rychlý přehled o stavu projektu a jeho kvalitativních vlastnostech. Bohužel v oblasti metrik znovu nástroj nenabízí požadovanou funkčnost, pouze částečně a to s nutností manuálního sestavení metrik a vizualizace. Dále nenabízí možnost pokročilejší definice vlastních metrik.

## 5.4 ReQtest

ReQtest je nástroj pro správu požadavků, testů a sledování chyb, založený na serverovém řešení. Skládá se z několika modulů – modul požadavků, modul testů, agilní nástěnka, modul chyb a modul reportů. ReQtest poskytuje komplexní možnost správy softwarového i jiného projektu a tím napomáhá týmům dojít do zdárného cíle projektu [23].

### 5.4.1 Analýza nástroje ReQtest

První stránka, kterou uživatel uvidí, je přizpůsobitelná nástěnka, na které si uživatel nastaví přehled vybraných informací a reportů z projektu, ukázku takové stránky lze vidět na obrázku 15. Mezi hlavní funkcionalitu nástroje patří správa požadavků, které bude rozebírána v následující podkapitole. Další funkcionalitou je správa testů, která je speciálně navržena pro profesionály v oblasti zajištění kvality (anglická zkratka QA).




Obrázek 15: Nástěnka uživatele v nástroji ReQtest

Správa testů nabízí vytváření testovacího plánu, kdy je možné naplánovat testovací kola a přiřadit je uživatelům. Jednotlivé testovací kola se skládají z více testovacích případů, čímž lze například otestovat celý případ užití, složený z více úkonů (testovacích případů) najednou. Výsledek běhu testovacího kola hodnotí uživatel a toto hodnocení je viditelné v grafické formě na

hlavní stránce modulu s výpisem všech testovacích kol a také ve formě koláčového grafu v daném testovacím kole. Popis testovacího případu je standardní a v případě, že by nedostačoval, nabízí nástroj i možnost definice vlastních vlastností. V případě neúspěšného výsledku testu nabízí nástroj přímé vytvoření a napojení chyby do modulu chyb. Součástí modulu správy testů je také rozhraní pro kontrolní seznam, kde uživatel vytvoří zvolený seznam úkolů, kde poté přehledně vidí, které další testy a úkoly zbývá provést a které má již hotové. Samozřejmostí je napojení úkolů na úspěšné provedení testu.

Modul sledování chyb nabízí rozšířenou práci s chybami. Chyby mohou být vytvořeny několika způsoby, první byl již zmíněn výše – přímý import při běhu testů, druhý způsob je manuální vytvoření a třetí způsob je import. Moduly pro chyby a testování nabízí import dané entity z CSV souboru nebo možnost přímého napojení na rozhraní nástroje JIRA. Mírnou nevýhodou může být, že nástroj ReQtest nenabízí integraci s dalšími nástroji, ani jinou možnost importu dat. Chyby mohou být navázány na požadavky nebo testovací běhy, díky tomu je možné vysledovat vznik chyby a ohlídat její opravení nebo vyřešení.

Samozřejmostí pro všechny moduly a jejich artefakty je vedení podrobné historie úprav a změn pro každý jednotlivý artefakt, ať už se jedná o požadavek, testovací případ nebo chybu. Další nepostradatelnou funkcionalitou je také vysledovatelnost pomocí odkazů na jiné artefakty. Tyto odkazy lze navázat vždy, například mezi požadavek a testovací případ, který splnění požadavku testuje.

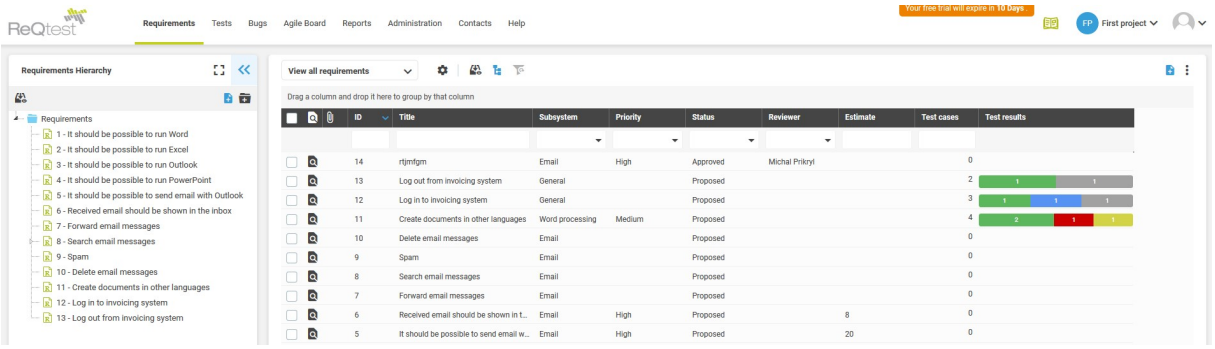
Další užitečnou funkcionalitou/modulem je agilní nástěnka. Agilní nástěnka nabízí přímočarý pohled na posun projektu. Jedná se o obdobu Kanbanu, kdy je možné vytvářet poznámky a ty si přetahovat mezi stavy. Agilní nástěnka je sdílená mezi uživateli, mají k ní tedy přístup všichni uživatelé přiřazení k projektu, čímž se agilní nástěnka stává místem komunikace a spolupráce mezi uživateli. Tento modul je primárně určen pro vývoj agilními metodami, ale je možné jej vhodně využít i pro všechny ostatní metody vývoje.

### 5.4.2 Správa požadavků v nástroji ReQtest

Správa požadavků v nástroji ReQtest nabízí velmi propracovanou agendu, vhodnou i pro náročnější projekty. Titulní stránka modulu požadavků zobrazuje výčet všech požadavků formou tabulky i s nejdůležitějšími atributy. Požadavky je možné vytvářet a zobrazovat ve stromové struktuře, čímž lze vymodelovat návaznosti mezi požadavky. Přehled stromové struktury požadavků je také zobrazen na titulní stránce, viz obrázek 16, kde je možné vidět výčet požadavků a zobrazenou stromovou strukturu.

Každý požadavek má dostačující množství atributů pro zachycení všech vlastností a stavů. K požadavku je možné připojit přílohy (dokumenty, obrázky aj.) a přidávat komentáře. Detail požadavku obsahuje čtyři záložky, první z nich je detail, kde jsou zobrazeny veškeré atributy požadavku, uživatelské komentáře a připojené přílohy. Druhou záložkou jsou odkazy, které představují připojené artefakty z projektu, například testovací případy, chyby atp. Třetí záložkou jsou požadavky navazující na prohlížený požadavek, tyto požadavky vytváří dříve zmiňovanou

stromovou strukturu. Poslední záložkou je aktivita, zde je možné vidět veškeré úpravy a kroky, které byly s požadavkem provedeny.




Obrázek 16: Výčet všech požadavků v nástroji ReQtest

Nástroj umožňuje generovat reporty ve formě koláčového nebo sloupcového grafu. Reporty představují grafickou reprezentaci metrik, které napomáhají snadnému zjištění stavu projektu a jeho artefaktů. Reporty jsou variabilní a je možné je generovat pro požadavky, testy a chyby. Variabilní report znamená to, že lze měnit vlastnost artefaktu, která rozděluje vybrané artefakty do sloupců/výsečí v grafu. Lze tedy požadavky v grafu rozdělit například dle priority a dále dle statusu, jak je ukázáno na obrázku 17. Výhodou je také možnost vygenerování grafu do obrázku formátu jpeg, png nebo dokumentu PDF. Možnost vizualizace je sice větší než u jiných zkoumaných nástrojů, ale požadovaných vlastností stále nedosahuje. Chybí zde například možnost definice uživatelských metrik a následná agenda.


Obrázek 17: Metrika vyjádřená pomocí grafu v nástroji ReQtest

### 5.4.3 Zhodnocení nástroje ReQtest

Nástroj ReQtest je znovu velmi vydařeným zástupcem nástrojů pro správu projektů, který má jednoduché, ale plně dostačující uživatelské rozhraní. Jako mírné nedostatky lze vidět například chybějící možnost integrace více služeb a velmi malé možnosti v oblasti souborového importu

a exportu. Další možnou nevýhodou je chybějící agenda úkolů, kde by bylo možné rozdělovat úkoly zaměstnancům a připojovat úkoly k artefaktům vývoje.

Jako výhodu lze považovat jednoduché ovládání, možnost generování grafických výstupů a metrik, které ale bohužel nejsou dostačující definovaným požadavkům na rozšíření IBM Jazz. Nižší cena licence a provozování nástroje než u konkurence, má v tomto případě za následek nepatrně nižší počet a rozsah možností funkcí.

## 5.5 Další analyzované nástroje a aplikace

V této podkapitole jsou krátce popsány nástroje, které byly analyzovány, ale jejich funkcionalita nebyla natolik kvalitní nebo rozsáhlá, že jejich použití pro správu požadavků, generování přehledných měření a metrik a ve výsledku rozšíření platformy IBM Jazz nebylo možné ani v základní míře.

### 5.5.1 Process street

Nástroj Process street slouží ke správě procesů, týmové spolupráci a vytváření jednoduchých „todo“ formulářů. Nabízí jednoduché a uživatelsky přívětivé grafické rozhraní webové aplikace s množstvím předdefinovaných šablon pro různé případy užití. V rámci Process street lze efektivně zachytit business procesy spolu s grafickým znázorněním, slovním popisem a dalšími doplňujícími atributy. Dále lze pohodlně vytvářet kontrolní dotazníky, seznamy práce a jiné podobné druhy dokumentů [24].

Správa požadavků v rámci Process street je jako jiné funkce velmi přívětivá. Bohužel není natolik pokročilá jako u jiných zkoumaných nástrojů, a to hlavně z důvodu, že tato funkčnost není v tomto nástroji primární. Chybí zde například samostatná sekce pro požadavky a není také možné vizualizovat v čase metriky týkající se požadavků. Zkrátka požadavky lze spravovat, ale pouze v rámci jednoho dokumentu, ne v globálním nebo projektovém měřítku.

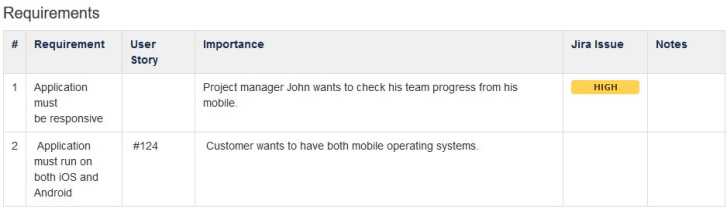
### 5.5.2 Confluence a JIRA Core

Jak Confuence, tak i JIRA Core, patří do rodiny business nástrojů firmy Atlassian. Výhodou těchto nástrojů je právě propojení v rámci jednoho poskytovatele služeb. Další nespornou výhodou je i to, že jsou nástroje firmy Atlassian dostupné přes webové rozhraní, tudíž není nutné nástroj instalovat ani nastavovat – vše běží v cloudovém prostředí vydavatele. V obou případech se jedná o nástroje komerční a tudíž i placené, které ale umožňují využít třicetidenního zkušebního období, kde je možné si vyzkoušet plnou verzi vybraného nástroje, potažmo více nástrojů, zdarma.

Principem nástroje Confluence je otevřený a sdílený pracovní prostor, kde mohou uživatelé sdílet a spojovat nápady a informace, aby při práci dosáhli toho nejlepšího možného výsledku. Na výběr je velké množství předpřipravených šablon pro různorodé úkoly a procesy. Lze plánovat

projekt, zapisovat poznámky, vytvářet marketingový plán, psát blog a v neposlední řadě také shromažďovat produktové požadavky [25].

Správa požadavků v rámci Confluence je možná, ale pouze na úrovni jednotlivých stránek (dokumentů), jedná se tedy spíš o formu poznámek, viz obrázek 18, kterou nelze nijak blíže třídit a spravovat než v rámci daného dokumentu. Agenda metrik či měření kvality tudíž již z podstaty tohoto nástroje není dostupná. Tento nástroj může najít uplatnění v brzkých fázích vývoje, kdy slouží jako pokročilý a strukturovaný poznámkový blok s možností paralelního přístupu více uživatelů.



Requirements

| # | Requirement | User Story | Importance | Jira Issue | Notes |
|---|---|---|---|---|---|
| 1 | Application must be responsive | | Project manager John wants to check his team progress from his mobile. | HIGH | |
| 2 | Application must run on both iOS and Android | #124 | Customer wants to have both mobile operating systems. | | |

Obrázek 18: Příklad zápisu požadavků v nástroji Confluence

JIRA Core je nástroj pro správu firemních projektů. Slouží jako místo, kde je možné přidělovat požadavky uživatelům (zaměstnancům) a kde je možné sledovat vývoj, stav a změny požadavků v reálném čase. Nástroj nabízí přehledné grafy, u kterých je možné nastavit periodické generování a zasílání daným uživatelům pro rychlou kontrolu změn v životním cyklu požadavku. Díky možnosti zobrazení změn a úprav požadavků je možné sledovat aktuální stav přehledně na jednom místě bez nutnosti svolávat meeting nebo rozesílat e-maily.

Požadavkem je v kontextu JIRA Core primárně myšlen úkol pro zaměstnance, ale díky podobnosti prováděných úkonů u správy požadavků a úkolů, lze tento nástroj v mírně omezené míře použít i pro správu zákaznických požadavků projektu. Jednotlivým požadavkům (úkolům) lze přidělovat atributy shodné s atributy požadavku, vytvářet a zobrazovat sestavy, jinými slovy výstupy – grafy, kde je možné získat přehledně informace o stavu zpracování požadavků a jiných metrikách. Každý požadavek je možné přidělit zaměstnanci, který bude například zodpovědný za jeho implementaci a otestování, tímto je tedy i možné zobrazit graf s informací, kolik požadavků je pokryto/zpracováno zaměstnanci a kolik ne.

I v případě nástroje JIRA Core není dostupná agenda metrik, ať už z pohledu definice a správy vlastních metrik, tak ani z pohledu pokročilejší možnosti vytváření grafů.

### 5.5.3 Easy Projects

Easy Projects [27] je software pro správu projektu vyvíjený firmou Logic software. Nástroj je založen na webové aplikaci, která umožňuje online správu projektu pomocí webového prohlížeče. Většinou je implementován jako online SaaS, což poskytuje úplné softwarové řešení pronajímané od poskytovatele aplikace přístupné přes internet [28]. Druhou možností získání a provozování nástroje je hostování na vlastním serverovém řešení. Easy projects také nabízí přístup pomocí

mobilní aplikace dostupné pro operační systémy Android a iOS, kde je možné v omezené míře spravovat projekt a komunikovat s ostatními členy týmu pomocí integrovaného chatu.

Bohužel Easy Project je spíše nástrojem primárně určeným ke správě projektu, což představuje činnosti jako vytváření a přiřazování úkolů/aktivit, administraci aktivit, generování časového průběhu projektu a dalších grafických výstupů. Z oblasti správy požadavků je zde jen velmi málo funkcí, tudíž je tento nástroj nedostačující požadavkům pro pokročilý softwarový proces a generování metrik procesu. Jedná se o nástroj podobný nástroji JIRA Core, s tím rozdílem, že nabízí méně funkcionalit v rámci ekosystému společnosti, než nabízí JIRA, potažmo její tvůrce Atlassian.

## 5.6 Komplexní zhodnocení všech zkoumaných nástrojů

Většina zkoumaných nástrojů, vyjma nástrojů obsažených v předchozí podkapitole 5.5, poskytuje poměrně obstojné prostředí pro správu softwarového projektu, horší už je ale stav v oblasti metrik a měření kvality, kdy většina nástrojů neposkytuje dostatečnou funkcionalitu. Některé nástroje poskytují více funkcí než ostatní, což je ale ve většině případů vykoupeno vyšší pořizovací cenou. Všechny zkoumané nástroje jsou placené, ale pro vyzkoušení poskytují plnou verzi nástroje v omezeném zkušebním období zdarma. Ceny se pohybují, v případě periodicky placených licenci, v řádech jednotek až desítek eur za uživatele, a v případě pevných licencí (např. desktopová aplikace nebo licence bez omezení počtu uživatelů) se cena pohybuje v rozmezí stovek eur.

Přehledné vyhodnocení analyzovaných nástrojů lze najít v tabulce 3, kde jsou vybrány nejdůležitější funkce z pohledu měření kvality softwaru a metrik. Funkce nástrojů jsou pro přehlednost tabulky popsány čísly. Význam čísel je následující:

1. podpora celého vývojového cyklu
2. webový přístup
3. integrace s podobnými nástroji
4. vysledovatelnost a propojení artefaktů
5. možnost rozšíření pomocí souborů nebo REST API
6. základní možnost vizualizace metrik
7. pokročilejší metriky a možnost definice vlastních metrik

| **Nástroj** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|
| IBM DOORS Next Generation | ano | ano | ano | ano | ano | ano | ne |
| codeBeamer | ano | ano | ano | ano | ano | ano | ne |
| Enterprise Architect | ano | ne | ano | ano | ano | ano | ne |
| SpiraTeam | ne | ano | ano | ano | ne | ano | ne |
| ReQtest | ne | ano | ne | ano | ne | ano | ne |

Tabulka 3: Přehled některých možností a funkcí zkoumaných nástrojů

V možnostech napojení a integrace jsou jednotlivé nástroje na rozdílných úrovních. Nástroje jako codeBeamer ALM nabízí velké množství napojení, což může být výhodou při integraci s již používanými nástroji. Obecně lze tvrdit, že čím větší oblíbenosti u uživatelů se nástroj těší, tím jsou také možnosti jeho integrace a komunikace s jinými nástroj větší. Méně používané nástroje nabízí menší množství propojení, například pouze s nejdůležitějšími nástroji trhu. V oblasti

generování reportů do formátu PDF nebo MS Office nabízí každý nástroj více či méně rozsáhlou funkcionalitu, ale tato vlastnost není z hlediska této práce natolik důležitá.

Z pohledu správy požadavků a s nimi spojených metrik a měření kvality, dopadly zvolené nástroje taktéž dobře. V oblasti správy požadavků poskytují vybrané nástroje plnou podporu prostředí, byť s nepatrnými rozdíly ve funkcionalitě a možnostech. Rozdíly lze například najít v oblasti vysledovatelnosti a s tím spojené možnosti propojení artefaktů mezi sebou, kdy některé nástroje nabízely větší možnosti propojení a také zobrazení návazností, například pomocí matice nebo jiného grafického prvku. Měření kvality a metriky jsou ve vybraných nástrojích také implementovány, ale většinou jen v grafické podobě a to navíc pouze staticky – je možné zobrazit pouze aktuální stav metriky, ne historický průběh jejího stavu. Pro zobrazení metrik je ve všech nástrojích využito grafu, ať už koláčového nebo sloupcového, kde je možné vybírat dle jakého atributu dané entity se bude v grafu seskupovat. Například v nástroji Enterprise Architect je nabízena možnost zobrazení pokrytí požadavků pomocí vztahové matice, kdy je využito reprezentace ve stylu mřížky.

Z prozkoumaných nástrojů vyplývá to, že ve všech případech bude nutné využít rozšiřujícího nástroje pro výpočet a případné následné zobrazení metrik. Žádný nástroj nemá agendu metrik natolik rozšířenou, aby naplnil požadavky na metriky vyplývající ze standardu Automotive SPICE. Ve většině nástrojů chybí možnost jakéhokoli definování uživatelských metrik, vždy nabízí pouze základní výpočty/data vzniklé pouze v danou chvíli, bez možnosti uložení a budoucího načtení nebo jakéhokoli dalšího nastavení. Z tohoto pohledu tudíž žádný nástroj není plně vhodný pro rozšíření platformy IBM Jazz.

Samotnou kapitolou je nástroj IBM Rational DOORS Next Generation, potažmo platforma IBM Jazz, která nabízí pravděpodobně nejrozšířenější paletu nástrojů a funkčností ze všech zkoumaných nástrojů/platforem, jedinou vadou na kráse se jeví uživatelské rozhraní, které je velmi nepřehledné a složité, což může být vedlejším produktem rozsáhlé funkcionality. Nevýhodou je také dostupnost dokumentace, které je sice velké množství, ale je nepřehledná a obsahuje neplatné a duplicitní informace.

Nejvhodnější nebo nejlepší nástroj je těžké zvolit, jelikož každý má své výhody a nevýhody a je hlavně na koncovém uživateli, který nástroj se pro jeho vybraný účel hodí a který zároveň také nejlépe využije. Nicméně jako nejvhodnější nástroj, při vynechání IBM Rational DOORS Next Generation, se jeví nástroj codeBeamer ALM. Nabízí podobné, v mnoha ohledech i lepší, uživatelské rozhraní jako IBM DNG, díky čemuž se i při prvotním kontaktu jeví jako přívětivá aplikace s intuitivním použitím. Mírnou nevýhodou pro codeBeamer ALM se zdá být nevelká možnost vizualizace metrik nebo již možnost jejich výpočtu, ale tato funkcionalita není ani u IBM DNG nijak valně rozvinuta. Nicméně, jak již bylo zmíněno, možnost uživatelsky definovat pokročilejší metriky a alespoň základní forma vizualizace nebyla v žádném analyzovaném nástroji, tudíž je ve všech případech žádoucí využít dalšího rozšiřujícího nástroje, které ale také na trhu neexistují nebo jsou využívány jako korporátní řešení, určené pouze pro interní použití a nejsou dále distribuovány komunitě dalších uživatelů.

# 6 Praktická část

Cílem praktické částí této diplomové práce je vytvořit prototypový nástroj, který umí definovat metriky, sledovat zadané metriky v čase a zobrazovat a generovat reporty. To vše v návaznosti na požadavky definované v nástroji IBM DOORS Next Generation, který je součástí platformy IBM Jazz. Metrikami je myšleno převážně metriky potřebné k naplnění standardu Automotive SPICE. Nedílnou součástí praktické části práce je taktéž instalace, konfigurace a následné provozování platformy IBM Jazz, která je nutná i jako zdroj dat pro testování vyvíjeného nástroje.

## 6.1 IBM Jazz

Důležitou součástí praktické části této práce je také platforma IBM Jazz. Bez dostupné instance by nebylo možné dostatečně porozumět problematice platformy IBM Jazz a poté vyvíjet a testovat návazný vyvíjený nástroj pro práci s metrikami. Bližší informace o platformě IBM Jazz lze nalézt v kapitole 4, kde je popsána převážně z pohledu uživatelského, to znamená možností správy požadavků. V této kapitole bude platforma přiblížena z pohledu správcovského nebo spíše vývojářského, v návaznosti na problematiku okolo vyvíjeného nástroje.

### 6.1.1 Instalace IBM Jazz

Prvním a hlavním úkolem pro dostupnost instance IBM Jazz je instalace. Jednotlivé kroky instalace spolu s odkazy na komplexnější popis problému a poznámkami autora se nachází v příloze B. Nyní budou rozebrány pouze hlavní úskalí instalace. Jak již bylo zmíněno, IBM Jazz je serverová platforma, proto je nutné disponovat vlastním serverem, který je možné plně ovládat – s možností „root“ oprávnění. Z tohoto důvodu byl zřízen na katedrálních serverech virtuální server s nainstalovaným operačním systémem Ubuntu verze 18.04.01 LTS Bionic Beaver, s dostatečným výpočetním výkonem a konfigurací dle minimálních hardwarových požadavků. Ačkoli při běhu aplikace lze místy pocítit pomalejší odezvu, je server v takovéto konfiguraci pro potřeby testování a seznámení s nástroji plně dostačující. Kompletní výčet systémových požadavků pro instalaci instance IBM Jazz (Collaborative Lifecycle Management 6.0.6) lze nalézt na webu [29].

| | |
|---|---|
| **Operační systém** | AIX, IBM i, Linux, MacOS, Windows, z/OS |
| **Procesor** | 4 jádra s frekvencí 2 GHz |
| **Operační paměť** | 16 GB |

Tabulka 4: Minimální softwarové a hardwarové požadavky platformy IBM Jazz [30]

V tabulce 4 jsou vypsány všechny operační systémy dostupné pro instalaci IBM Jazz, zatímco hardwarové požadavky jsou vztaženy ke konkrétnímu operačnímu systému a to Linux, viz [30]. V tabulce 4 jsou vypsány minimální hardwarové požadavky, které jsou i přesto v celku vysoké. V případě produkčního prostředí, ke kterému bude přistupovat více uživatelů najednou, jsou

doporučené hardwarové požadavky větší v závislosti na zatížení. Samozřejmostí je také rozdělení jednotlivých částí (nástrojů) platformy na různé servery, aby se minimalizovalo momentální zatížení [31].

Instalace platformy IBM Jazz vyžaduje několik prerekvizit, které je nutné nainstalovat ještě před samotnou instalací. Většina prerekvizit je již často obsažena v operačním systému. Další specifické prerekvizity lze získat stažením odpovídajícího instalačního balíčku, který již v sobě obsahuje programy a nástroje, nutné pro provoz nebo nabízí jejich dodatečné stažení při instalaci. Mezi takové programy například patří aplikační server WebSphere Liberty nebo databázový server Apache Derby [32].

Pro potřeby testování je aplikační server WebSphere Liberty ve verzi 18.0.0.1 dostačující volba. WebSphere Liberty je Java aplikační server, postavený na open source projektu Open Library. Mezi jeho výhody patří jednoduchá konfigurace, rychlý start a celková rychlost, dostačující a vhodná i pro méně náročné produkční prostředí. Nabízí také jednoduché propojení s mnoha užitečnými nástroji a aplikacemi, jako například Docker nebo Jenkins, které pospolu usnadňují například vývoj [33].

V případě databáze bylo na doporučení nutné využít jiné možnosti než s instalací dodávaný open source databázový server Apache derby. Doporučeným databázovým serverem je DB2, také od společnosti IBM. DB2 je databázový server určený pro celopodnikové použití, nabízí vysoký výkon, flexibilitu a spolehlivost [34]. Z důvodu zmíněných výhod a také toho, že jak databázový server DB2, tak i IBM Jazz jsou od stejného vydavatele, bylo tedy zvoleno DB2 ve verzi Enterprise Server Edition. Výčet všech databází podporovaných v IBM Jazz lze najít v tabulce 5. U všech zmíněných verzí jsou podporovány i další budoucí aktualizace dané verze, tudíž se není nutné obávat problémů například při bezpečnostní aktualizaci [30].

| **Databáze** | **Edice** | **Verze** |
|---|---|---|
| DB2 | Advanced Enterprise, Advanced Workgroup, Enterprise, Express, Workgroup | 10.5, 11.1.0, 11.1.1.1 |
| Microsoft SQL Server | bez omezení | 2014, 2016 |
| Oracle | Enterprise, Standard | 11g, 12c |

Tabulka 5: Databáze podporované platformou IBM Jazz [30]

Po instalaci nutných prerekvizit a doplňkových nástrojů je na řadě instalace samotné platformy Jazz, potažmo vybraných nástrojů, které budou využívány. Základní rozdělení a popis nástrojů se nachází v kapitole 4.1. Nástroje se dále rozdělují na množství menších nástrojů. Pro využití v oblasti správy požadavků a metrik není nutné instalovat všechny nástroje a tím zbytečně zabírat místo na disku a ještě více zatěžovat server. Výčet instalovaných nástrojů je následující:

- Základním nástrojem je Jazz Team Server (zkratka JTS), který poskytuje základní služby umožňující propojení všech ostatních aplikací, aby spolupracovaly jako jeden celek a tvořily logický server.
- Nástroj Change and Configuration Management (zkratka CCM), který poskytuje služby pro rozdělování úkolů, správu chyb, správu zdrojového kódu, plánování a automatizaci sestavovaní a publikování.
- Nástroj Requirements Management (zkratka RM), který nabízí služby pro vytváření, správu a sledování požadavků v rámci celého vývoje.
- IBM Jazz Reporting Service (zkratka RS), který se skládá z několika nástrojů – Lifecycle Query Engine (LQE), Data Collection Component (DCC) a Report Builder. Soubor těchto nástrojů dohromady tvoří způsob, jak jednoduše získat informace (data) z různých zdrojů v rámci nástrojů životního cyklu IBM Jazz.

Po úspěšné instalaci je nutné vytvořit databáze a nastavit jednotlivé nástroje. Podrobnější kroky instalace i s komentářem autora lze najít v příloze B.

### 6.1.2 Export dat z IBM Jazz

Jako zdroj pro vyvíjený prototypový nástroj slouží požadavky uložené v nástroji IBM DOORS Next Generation. Proto je před samotným exportem nutné vytvořit požadavky a jejich návaznosti do nástroje IBM DOORS NG. Požadavky je nutné přiřazovat do modulů, dle jejich zařazení/skupiny – například softwarové požadavky (SWRS) nebo hardwarové požadavky (HWRS), aby byl umožněn jednodušší export dat a také v neposlední řadě jsou takto dodržovány „best practices“ správy požadavků.

Samotných způsobů exportu je hned několik. Každý z nich bude krátce popsán a bude zdůvodněno, proč je či není právě tento způsob vhodný a proč byl či nebyl zvolen jako zdroj dat pro vyvíjený nástroj. V poslední podkapitole se nachází celkové zhodnocení všech zkoumaných možností exportu dat.

**6.1.2.1 Visual Studio klient (VS client) a Eclipse klient** Visual Studio klient nabízí způsob integrace v jazyce C#, potažmo .NET Frameworku, bohužel pro jeho nedostupnost (nemožnost stažení a zprovoznění) a pravděpodobně i neaktuálnost, nebylo možné tento způsob podrobněji prozkoumat. Tato knihovna/SDK se jeví jako ukončený projekt, který již ze strany IBM není vyvíjen, tudíž by minimálně nesplňoval požadavky aktuální verze IBM Jazz a pravděpodobně také požadavky na bezpečnost atp. Po prozkoumání popisu nabízených metod se ani nejeví jako dostačující a to z důvodu, že nenabízí přístup k požadavkům v požadovaném rozsahu. Nabízí tedy přístup pouze v základním rozsahu, kdy není jednoznačně jasné, zda je možné získat i podrobnější nebo dodatečně přidané specifické atributy požadavků [35].

Existuje také Eclipse klient (vytvořen v jazyce Java), který ale nebyl pro export dat relevantní, jelikož je prototypový nástroj vyvíjen v jazyce C#. Nicméně v tomto případě by měl být vývoj neukončen.

**6.1.2.2 Reportable REST API** Reportable REST API slouží pro získání dat nutných pro reporty, například o stavu projektu. Slouží k získání dat pouze z nástroje Change and Configuration Management (CCM), kde lze nalézt převážné informace o projektu a zdrojovém kódu – z oblasti správy požadavků zde data nejsou. Data se získávají pomocí GET požadavků přímo na nástroj CCM, kde je do parametru URL adresy zadáván dotaz na data (požadavek na filtrování výsledků) ve formátu podobném jazyku XPath nebo SQL [36]. Z důvodu, že toto API nenabízí získání informací týkajících se správy požadavků, ale pouze informací z nástroje CCM, nelze jej tedy využít pro exportování dat nutných pro vyvíjený prototypový nástroj.

**6.1.2.3 DOORS Next Generation Reportable REST API** DOORS Next Generation Reportable REST API je podobné API jako předchozí zmíněné s tím rozdílem, že nabízí získání dat z nástroje Requirement Management (RM), tudíž požadované data správy požadavků. Dalším rozdílem je to, že dotazovací parametry jsou mírně rozdílné a to hlavně z důvodu jiné domény dotazovaných dat. Dotazování probíhá znovu pomocí URL adresy instance IBM Jazz za pomocí identifikátorů [37]. Takovéto řešení exportu dat se jeví jako použitelné pro vyvíjený nástroj, ačkoli nabízí v některých případech složité sestavování dotazů. Dále nastává nutnost ukládání dalších údajů, jako identifikátory projektů, artefaktů nebo modulů.

**6.1.2.4 Klientské rozšíření pro Requirement Management (RM)** Klientským rozšířením se v kontextu IBM Jazz rozumí webový modul implementovaný v jazycích HTML a JavaScript, který rozšíří stávající řešení nástroje RM formou přídavného modulu. Takovýto modul lze po instalaci zobrazit v levé části uživatelského rozhraní nástroje v mini nástěnce nebo na hlavní nástěnce. Pro podporu takovýchto rozšíření nabízí platforma IBM Jazz API dostupné ve všech nástrojích přes skripty jazyka JavaScript. Pomocí API jsou dostupné rozličné data nástroje a to od artefaktů, spojení (link) až po uživatele a mnoho dalších [38]. Díky tomu lze vyvíjet nenáročné rozšíření, které rozšiřují základní funkcionalitu nástroje.

Klientské rozšíření tedy nabízí způsob, jak exportovat data mimo prostředí IBM Jazz. Nicméně již v prvopočátku se jeví několik překážek. Jednou z hlavních je to, že by uživatel musel rozšíření instalovat a také manuálně spouštět export dat. Další problém vzniká při vývoji, kdy není možné aplikaci testovat jinak, než ji zdlouhavě nahrát do daného nástroje a až poté spustit a otestovat. Další nevýhodou je nepřítomnost komplexní funkce API, která by vracela všechny artefakty (minimálně z jednoho modulu), což by případně bylo možné vyřešit pomocí jiných funkcí, které by ale mohly způsobit problémy při exportu – vysokou výpočetní i časovou náročnost. I přesto by vždy byla nutná součinnost uživatele při každém exportu, což je pro uživatele nekomfortní a to není brána v potaz i instalace rozšíření s množstvím potencionálních prostorů

pro chybné kroky. Z výše zmíněných důvodů je tedy klientské rozšíření nevyhovující pro export dat do vyvíjeného nástroje.

#### 6.1.2.5 Requirements Interchange Format (ReqIF)

Requirements Interchange Format je speciální XML formát určený k výměně (ve většině případů softwarových) požadavků mezi různými nástroji a systémy. Standard definující tento formát je spravován konsorciem OMG a poslední verzí je verze 1.2 z července 2016 [39]. IBM DOORS Next Generation nabízí možnost importu a exportu požadavků pomocí souborů ve formátu ReqIF. Export dat probíhá tak, že uživatel vytvoří definici, dle které se formátují data ve výsledku exportu. Tento předpis je poté nutné předat opačné straně – příjemci dat, která dle tohoto předpisu získá informace z přijatého souboru. Do exportovaného souboru je možné zahrnout většinu dostupných datových struktur včetně odkazů mezi požadavky. Naopak nelze exportovat komentáře, revize nebo historie auditu [40].

Tento způsob exportu dat sám o sobě není vhodný a to z důvodu, že vyžaduje součinnost uživatele. Uživatel by musel manuálně zajišťovat přesun dat mezi nástrojem IBM DOORS Next Generation a cílovým vyvíjeným nástrojem pro generování metrik. Tento proces je možné automatizovat pomocí Rational DOORS Next Generation ReqIF API, které poskytuje rozhraní pro import a export ReqIF souborů a možnost vytvoření předpisu dat [41]. Díky ReqIF API je tedy možné tento způsob exportu ve vyvíjeném nástroj teoreticky využít, pouze s malou nevýhodou – je nutné nastavit uživatelské práva pro export dat.

#### 6.1.2.6 Open Services for Lifecycle Colaboration (OSLC)

Open Services for Lifecycle Colaboration, krátce popsané v kapitole 4, poskytuje přístup k informacím nástrojů platformy Jazz pomocí RESTful rozhraní. Hlavním konceptem OSLC je to, že web je definován, kromě formátu HTML, také ve formátu RDF, čímž je možné získávat data z různých serverů jednotným způsobem pomocí protokolu HTTP. Hlavní výhodou je jednotný přístup k datům, což umožňuje znovupoužití již existujícího způsobu připojení z jednoho serveru podporujícího OSLC na jiný pouze s minimálními úpravami [42].

OSLC, potažmo jeho vydavatel OASIS, nabízí SDK/knihovny pro použití v různých programovacích jazycích jako Java, C# (.NET) a JavaScript, které podporují vytváření nových služeb implementujících OSLC nebo napojení na již existující služby implementující OSLC. Poskytovatelé služby OSLC pro správu požadavků musí splňovat podmínky dané specifikací, jako například podmínky na zabezpečení (HTTP Basic nebo OAuth), přístup k datům (REST API), reprezentaci dat (XML, JSON, HTML, RDF) nebo práci s daty (CRUD operace - HTTP GET, PUT, POST), čímž je zajištěno znovupoužití kódu v maximální možné míře při nutnosti napojení na další server poskytovatele služeb OSLC [43]. Podrobnější informace o specifikaci OSLC v rámci IBM Jazz lze najít v oficiální nápovědě IBM Jazz zde. Ukázka požadavku ve formátu JSON při využití OSLC REST API se nachází ve výpisu 1.

```
{
    "prefixes": {
        "dcterms": "http:\/\/purl.org\/dc\/terms\/",
        "oslc": "http:\/\/open-services.net\/ns\/core#",
        "oslc_cm": "http:\/\/open-services.net\/ns\/cm#",
        "oslc_cmx": "http:\/\/open-services.net\/ns\/cm-x#",
        "rdf": "http:\/\/www.w3.org\/1999\/02\/22-rdf-syntax-ns#",
        "rtc_cm": "http:\/\/jazz.net\/xmlns\/prod\/jazz\/rtc\/cm\/1.0\/",
        "rtc_ext": "http:\/\/jazz.net\/xmlns\/prod\/jazz\/rtc\/ext\/1.0\/"
    },
    "dcterms:contributor": {
        "rdf:resource": "https:\/\/localhost:9443\/jazz\/oslc\/users\/
            _BUIEYZ5bEeCZ1JTYjwz0UQ",
        "rdf:type": [
            {
                "rdf:resource": "http:\/\/xmlns.com\/foaf\/0.1\/Person"
            }
        ]
    },
    .....
}
```

Výpis 1: Ukázka požadavku při využití OSLC ve formátu JSON

V počáteční fázi vývoje nebyl tento způsob exportu využit, jelikož není zcela jasné, zda je vhodný pro požadovaný případ užití ve vyvíjeném nástroji, hlavně pro jeho komplexnost a z toho plynoucí složitost. Proto je ponechán prostor pro následnou práci kolegů v dalších letech, kteří mohou tento způsob exportu využít a rozšířit či nahradit vybraný způsob exportu.

**6.1.2.7 IBM DOORS Next Generation API** IBM DOORS Next Generation API (někdy psáno pod zkratkou IBM DNG API) nabízí možnosti, které nenabízí služby OSLC, tudíž se stává pomyslným rozšířením služeb OSLC. Příkladem rozšíření funkcionality je například aktualizace některých hodnot, přidávání a odebírání artefaktů z/do modulů nebo také získání artefaktů spolu se strukturou vztahů mezi artefakty a odkazem na navazující artefakt (vztah rodič/potomek). Nabízí tedy podobný způsob dotazování a stejné možnosti formátu dat jako OSLC [44]. Jako samostatný zdroj dat by tento způsob vhodný nebyl, ale ve spojení s OSLC se již jeví lépe, tudíž použitelný pro vyvíjený nástroj.

#### 6.1.2.8 Jazz Reporting service (RS)

Jazz Reporting service je základní reportovací nástroj v rámci platformy IBM Jazz. Skládá se ze tří komponent a to:

- Report Builder – nabízí jednoduché rozhraní pro vytváření vlastních reportů a jejich následné ovládání a nastavování.
- Lifecycle Query Engine (LQE) – úkolem této komponenty je indexovat data projektů pro vytváření reportů.
- Data Collection Component (DCC) – sbírá data z projektů do datového skladu (v originále Data Warehouse), které se poté využívají jako zdroj pro vytváření reportů.

Komponenta Report Builder je tedy primárním zdrojem dat, kde lze buď vytvořit nebo naimportovat předpisy pro reporty. Jako zdroj dat pro Report Builder slouží buď Lifecycle Query Engine nebo Data Collection Component a záleží na požadavcích uživatele, jaký druh dat potřebuje získat. Report Builder nabízí export do různých formátu jako HTML, PDF, Microsoft Word nebo XML (určeno pro Microsoft Excel), jehož příklad lze vidět ve výpise 2. Pro export dat do vyvíjeného nástroje se jeví jako použitelný export ve formátu XML [45].

```
<results xs:schemaLocation="https://158.196.141.113/rs/query/1/dataservice/ns
    https://158.196.141.113/rs/query/1/dataservice/xsd">
   <result>
      <PROJECT_NAME>Test project</PROJECT_NAME>
      <REQUIREMENT_TYPE>Requirement</REQUIREMENT_TYPE>
      <NAME>HWRS</NAME>
      <REFERENCE_ID>4</REFERENCE_ID>
   </result>
</results>
```

Výpis 2: Ukázka výstupu komponenty Report Builder ve formátu XML

Lifecycle Query Engine je vhodný pro méně rozsáhlé projekty, které neobsahují velké množství záznamů – například artefaktů, jelikož pro svůj běh (indexaci) vyžaduje vysoký výpočetní výkon. Tudíž je nutné vlastnit výkonný server, aby nedošlo k nežádoucímu zpomalení ostatních nástrojů a aplikací běžících na stejném serveru. Z tohoto důvodu je tedy vhodné použít jako zdroj dat Data Collection Component. Data Collection Component zajišťuje samotný export dat ze všech nainstalovaných nástrojů platformy IBM Jazz do databáze Data Warehouse, ze které poté čerpá Report Builder data pro reporty – export dat [46]. Export dat probíhá pomocí periodicky se spouštějících úloh, které je také možné spouštět manuálně ve webovém rozhraní. Rychlost zpracování je vylepšena, díky paralelnímu běhu úloh [45].

Report Builder nabízí flexibilní způsob exportu dat z IBM Jazz a to díky možnosti definice vlastních reportů, možnosti exportu stávajících a importu nových definic reportů. Díky tomu

je možné přistoupit k jakýmkoli informacím pouze za cenu definování nebo importu předpisu reportu. Přístup k výsledkům reportu je také přívětivý a to díky možnosti výsledného formátu XML, který lze programově zpracovat a získat z něj potřebné data. Jedinou mírnou nevýhodou se jeví to, že export dat do datového skladu probíhá pouze jednou denně, což lze přenastavit tak, aby export probíhal častěji a tím bylo dosaženo získání co nejaktuálnějších dat.

**6.1.2.9 Zhodnocení možností exportu dat** IBM Jazz, potažmo nástroj IBM DOORS Next Generation, nabízí množství různých možností exportů dat. Některé z nich jsou z pohledu přístupu novější/modernější (například REST API), jiné starší a již pravděpodobně neaktualizované, jako například .NET Client. U některých možností bylo hned z počátků patrné, že jsou nevyhovující pro daný účel, ale z důvodu komplexnosti průzkumu jsou zde i tyto možnosti ponechány. Jednotlivé možnosti byly podrobněji hodnoceny již v rámci vlastního popisu. V tabulce 6 jsou jednotlivé možnosti exportu dat přehledně porovnány mezi sebou dle následujících vlastností:

1. vhodné pro využití při exportu dat
2. vyžaduje při exportu pravidelné zapojení uživatele
3. vyžaduje počáteční nastavení IBM Jazz
4. složitost a náročnost řešení

| **Přístup** | **1** | **2** | **3** | **4** |
|---|---|---|---|---|
| Visual Studio klient | ne | — | — | — |
| Reportable REST API | ne | — | — | — |
| DOORS Next Generation Reportable REST API | ano | ne | ne | ano |
| Klientské rozšíření pro Requirement Management | ne | ano | ano | ano |
| Requirements Interchange Format | ano | ne | ano | ne |
| Open Services for Lifecycle Colaboration | ano | ne | ne | ano |
| IBM DOORS Next Generation API | ne | ne | ne | ano |
| Jazz Reporting service | ano | ne | ano | ne |

Tabulka 6: Přehled vlastností zkoumaných možností exportu dat

Ze srovnání v tabulce 6 vychází, že polovina nástrojů je (případně i s výhradami) vhodná pro export dat z IBM Jazz do vyvíjeného nástroje a druhá polovina ne. U prvních dvou možností exportu (Visual Studio klient a Reportable REST API) ani nebyly podrobněji zkoumány vlastnosti, jelikož bylo na první pohled patrné, že nejsou vhodné pro export dat v aktuálním případu užití. Klientské rozšíření pro Requirement Management by sice bylo možné využít, ale

za cenu zapojení uživatele, což není v tomto případě vhodné. Poslední nevyhovující možnost IBM DOORS Next Generation API samo o sobě vhodné není, ale jako doplněk k OSLC by bylo možné využít.

První vhodnou možností exportu dat je DOORS Next Generation Reportable REST API, které nabízí klasické REST API s možností automatizovaného získání dat, jedinou nevýhodou může být vyšší čas nutný k získání odpovědi, protože data je nutné získávat v některých případech z více zdrojů (databází). Další akceptovatelná možnost exportu Requirements Interchange Format nabízí při spojení s ReqIF API také automatizovanou možnost získání dat, jedinou nevýhodou je nutnost počátečního nastavení uživatelských práv pro export ReqIF souborů. Možnost exportu dat pomocí Open Services for Lifecycle Colaboration nabízí flexibilní řešení napojení pomocí REST API, které je plně automatizované a nevyžaduje zapojení uživatele, ani v případě počátečního nastavení. Poslední akceptovanou možností exportu je Jazz Reporting service, který nabízí poměrně snadno přístupné rozhraní pro získání dat, mírnou nevýhodou může být nutnost počátečního nastavení, ale po tomto kroku je již export dat plně automatický a nevyžaduje žádné akce.

Všechny čtyři vhodné přístupy je možné reálně využít, nicméně na doporučení vedoucího práce Ing. Svatopluka Štolfy, PhD. bylo využito v prvotní fázi vývoje nástroje možnosti Jazz Reporting service, která sice vyžaduje počáteční nastavení, ale svou další jednoduchostí předčí ostatní možnosti exportu. Dalším důvodem pro výběr bylo mimo jiné to, že takovéto řešení je využito v aktuálním nevyhovujícím řešení předchůdce vyvíjeného nástroje. Jak již bylo zmíněno, při dalším vývoji nástroje je doporučeno využít exportu dat pomocí služeb OSLC. Využití OSLC se přináší i možnost získání dat pro metriky z jiných zdrojů než jen IBM DOORS Next Generation, což lze splnit za předpokladu, že bude standard OSLC rozšířen do více nástrojů z oblasti správy požadavků a vývojáři jej plně dodrží.

## 6.2 Vize nástroje

Vizí je, aby nástroj přehledně zobrazoval metriky nutné pro naplnění standardu Automotive SPICE. Nástroj by měl být přístupný přes webovou aplikaci, aby byl zajištěn jednoduchý přístup. Měl by dále umožňovat základní práci s uživateli a členění uživatelů do rolí s různými uživatelskými právy a z toho plynoucími možnostmi v rámci aplikace. Jednotlivé metriky budou přiřazeny k projektu, kdy každá z nich bude vázána s vybranou skupinou uživatelů. Metriky projektu budou formou grafu zobrazeny na nástěnce, kde uživatel přehledně uvidí aktuální i historický stav projektu vyjádřený pomocí metrik. Hodnoty nutné pro generování metrik a grafů bude možné načítat přímo z nástroje IBM DOORS Next Generation, kdy bude import dat probíhat pomocí vygenerovaných XML souborů, které se automaticky stáhnou z vložené URL a dále zpracují do požadované podoby. Metriky mohou být různých typů a v budoucnu se mohou měnit v závislosti na změnách ve standardu ASPICE, proto je žádoucí, aby nástroj měl možnost definice nových metrik dle aktuálních požadavků standardu. Nejen metriky samotné se mohou měnit v závislosti na změnách standardu nebo trendu, proto je žádoucí, aby bylo možné v apli-

kaci měnit veškeré nastavení, týkající se metrik. Další vítanou funkcionalitou je možnost zasílání nebo zobrazení upozornění na klesající trend metriky v čase nebo na nízkou hodnotu metriky a podobné nežádoucí jevy.

## 6.3 Základní funkcionalita nástroje

Základní funkcionalitu nástroje lze rozdělit do tří částí – uživatelská, projektová a aplikační. Do uživatelské spadá funkcionalita jakou je správa uživatelského účtu nebo správa společnosti. Do projektové funkcionality spadá správa metrik a projektů. Poslední část funkcionality – aplikační, obsahuje funkcionalitu nutnou k podpoře běhu aplikace. Veškerou funkcionalitu ve formě diagramu případu užití lze vidět v přehledné formě na obrázku 19. Uživatelé v nástroji mohou nabývat čtyř rolí, od toho se také odráží dostupnost funkcionality pro danou roli v návaznosti na míru odpovědnosti uživatele.


Obrázek 19: Diagram případu užití vyvíjeného nástroje

Úplně první rolí je Neregistrovaný uživatel. Takový uživatel má pouze možnost registrace do nástroje a při té příležitosti také vytvoření jeho vlastní společnosti. Přístup k jiným funkcionalitám je podmíněn registrací, tudíž aby uživatel mohl využít další funkcionalitu nástroje, musí mu být přiřazena určitá vyšší role.

Základní rolí registrovaného uživatele je role Uživatel (User). Uživatel v roli User má samozřejmě dostupnou funkcionalitu týkající se uživatelského účtu, což zahrnuje editaci vlastního účtu a změnu hesla. Pokud je uživatel členem společnosti a je zároveň přiřazen k projektu, má možnost přístupu k nástěnce projektu s metrikami. Dále může exportovat grafy metrik z nástroje nebo aktualizovat hodnoty metrik. Pokud uživatel není přiřazen ke společnosti, může vytvořit vlastní společnost, čímž je jeho uživatelskému účtu automaticky přiřazena administrátorská role s názvem Admin.

Uživatelská role, určená pro pokročilejšího registrovaného uživatele, s názvem Admin nabízí přístup ke stejné funkcionalitě jako uživatelská role User, k tomu je přidána další funkcionalita taková, aby uživatel mohl plně spravovat svou společnost a její projekty, potažmo s projektem související metriky. Každá společnost (její uživatel) má možnost definice vlastních předpisů metrik, které mohou být přístupné jak jen jeho společnosti, tak i všem ostatním společnostem v rámci nástroje. Vytváření, editace a mazání projektu je také přístupné pouze uživateli v roli Admin a výše, jelikož se již jedná o rizikovější operace, které musí vykonávat pouze zodpovědný uživatel. Uživatel v roli Admin má také přístup k funkcionalitě administrace projektu, což zahrnuje přidávání a odebírání uživatelů společnosti k projektu a operace s metrikami projektu, jako přidávání, mazání a editace. V neposlední řadě se uživatel v roli Admin stará o svou společnost, přidává nové uživatele, maže uživatele, odebírá uživatelské práva anebo naopak i přidává.

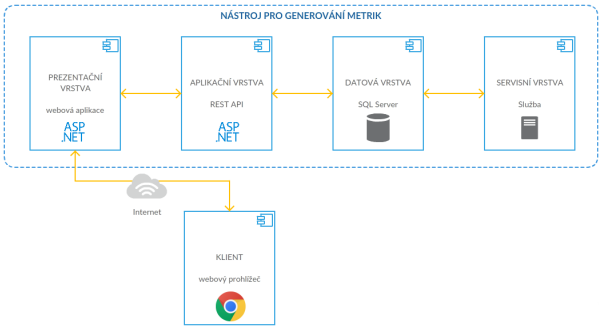
Nejrozšířenější přístup k funkcionalitě nástroje má uživatel v roli SuperAdmin (zahrnující role User a Admin), takovýto uživatel má dokonce přístup ke kompletní funkcionalitě nástroje. Uživatelská role SuperAdmin obsahuje možnost přidávání, odebírání a editace obecných typů metrik, dále možnost přidávání, odebírání a editace ovlivněných oblastí vývoje. Důležitá je také administrace verzí standardu Automotive SPICE, což zahrnuje možnost přidávání, odebírání a editace verzí standardu a z nich vyplývajících procesů. V neposlední řadě má uživatel v roli SuperAdmin dostupnou funkcionalitu správy aplikace – globálního nastavení aplikace a kontrolu chybového logu s možností řešení nastalých chyb.

## 6.4 Architektura nástroje

Po analýze podobných nástrojů a stávajících řešení byly vyhodnoceny klady a zápory jednotlivých řešení, z toho vyplynuly dvě možnosti architektury aplikace. První možností byla klasická klientská aplikace (například desktopový program) a druhou možností webová aplikace, neboli podnikové řešení. Klady podnikového řešení převážily klady klasické aplikace a tudíž bylo zvoleno řešení pomocí webové aplikace, a to z důvodu přístupu nezávislého na operačním systému uživatele, jednodušší správě a aktualizaci aplikace, a v neposlední řadě také to, že uživatel nemusí nic instalovat, kromě webového prohlížeče, který ve valné většině případů již nainstalovaný má.

Nástroj se skládá ze čtyř hlavních částí/komponent, jak lze vidět na obrázku 20, kde je znázorněna architektura nástroje. První a nejdůležitější částí je datová vrstva, která je blíže popsána v následující podkapitole 6.4.1. V Datové vrstvě se nachází persistentní úložiště ve formě relační

SQL databáze, kde jsou uloženy veškeré uživatelské i aplikační data nutné pro běh aplikace. Další neméně důležitou částí nástroje je aplikační vrstva, kterou tvoří REST API, zajišťující komunikaci mezi prezentační a datovou vrstvou, tato vrstva je přiblížena v podkapitole 6.4.2. Třetí částí je prezentační vrstva, se kterou se dostává do kontaktu koncový uživatel nástroje a v níž provádí procesní úkony, kontroluje stav projektu atd. Prezentační vrstvou se zabývá podkapitola 6.4.3. Poslední součástí nástroje je servisní vrstva, zajišťující automatickou aktualizaci hodnot metrik z platformy IBM Jazz. Servisní vrstva je krátce popsána v podkapitole 6.5.




Obrázek 20: Návrh architektury nástroje

Jako nepřímou součást nástroje lze také brát platformu IBM Jazz, potažmo její nástroj IBM DOORS Next Generation, na který se servisní vrstva, konkrétně služba, vyvíjeného nástroje v pravidelných intervalech připojuje a čerpá z něj data nutné pro generování a zobrazení metrik. Tato část není v návrhu architektury na obrázku 20 zahrnuta. O IBM Jazz pojednává kapitola 4 a podkapitola 6.1.

### 6.4.1 Datová vrstva

Datová vrstva, neboli databáze, se zaměřuje na ukládání uživatelských dat. Pro persistentní ukládání dat je využito relační databáze a to Microsoft SQL Server 2017, která je krátce popsána v podkapitole 6.4.1.1. V následující podkapitole 6.4.1.2 jsou popsány jednotlivé entity databázového modelu.

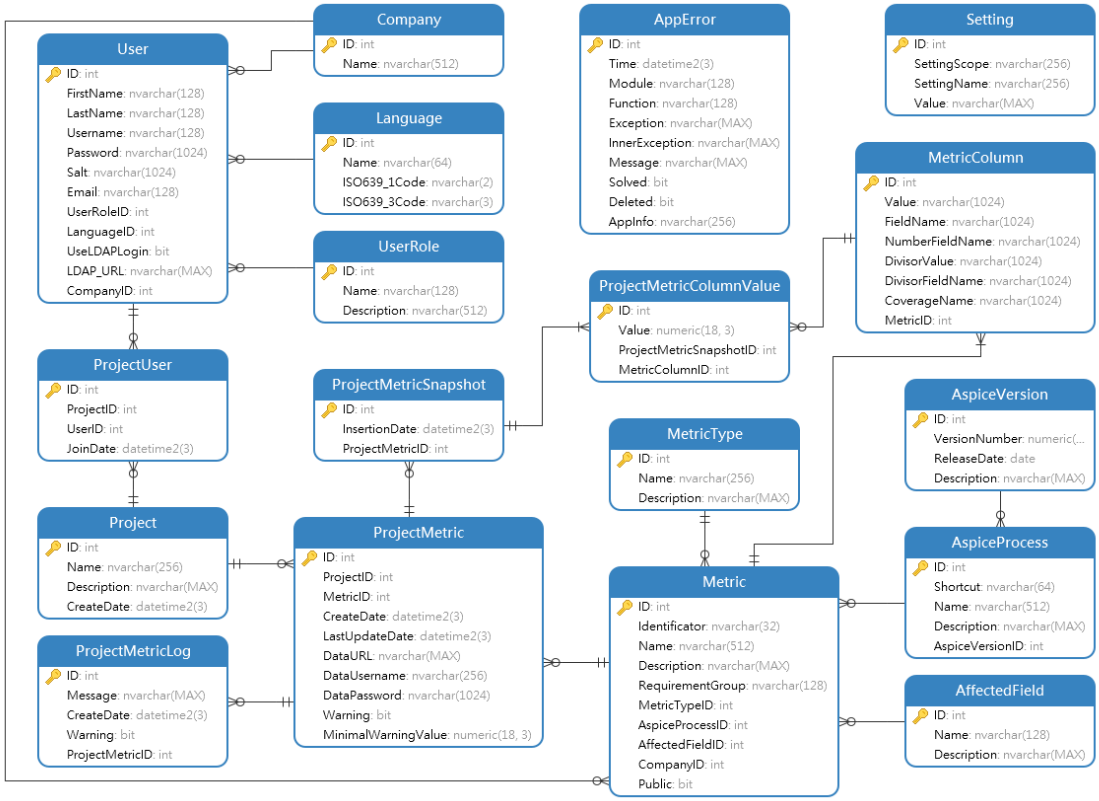
**6.4.1.1 Microsoft SQL Server 2017** Microsoft SQL Server je relační databázový systém vyvíjený společností Microsoft. Tento databázový systém je dostupný pro operační systémy Linux i Microsoft nebo také jako dnes moderní kontejner Docker. Nabízí vysoký a škálovatelný výkon i pro kritické aplikace a nástroje. Dle National Institute of Standards and Technology (NIST) se také jedná o jeden z nejlepších databázových systémů, co se týče míry zranitelnosti prostředí databáze [47].

Microsoft SQL Server 2017 byl zvolen z toho důvodu, že jeho provázání s jazykem C# (z důvodu stejného tvůrce – Microsoft) je vysoké. Druhým důvodem je velká zkušenost autora z tímto databázovým serverem. Pro potřeby ukládání dat je využito verze Microsoft SQL Server

2017 z října 2017, pro ni je také databáze nástroje vyvinutá, ale je očekáváno, že i nová verze Microsoft SQL Server 2019 bude zpětně kompatibilní s předchozími verzemi, mimo některé změny, jako například u dřívějších vydání [48]. Microsoft SQL Server nabízí mnoho edicí, od kterých se poté odvíjí možnosti výkonu databáze. Pro potřeby vyvíjeného nástroje je dostačující i základní verze Express. Verze Express byla využita také jako testovací a produkční databázový server, provozovaný skrze službu Microsoft Azure.

#### 6.4.1.2 Popis datové vrstvy (databáze)

Databázi lze rozdělit do čtyř logických segmentů a to uživatelského segmentu, projektového segmentu, segmentu metrik a segmentu správy aplikace. První tři segmenty jsou spolu navzájem propojené, jen segment správy aplikace je samostatný – nemá návaznosti na ostatní entity. ER diagram databáze je vyobrazen na obrázku 21. ER diagram databáze ve formě obrázku, soubor s databázovým modelem pro nástroj Navicat for SQL Server a všechny potřebné databázové skripty se nachází v příloze práce a v git repozitáři, viz kapitola 6.6.2.




Obrázek 21: ER diagram databáze nástroje

Uživatelský segment se skládá z pěti entit. Hlavní entitou je entita *User*, která představuje uživatele – jejich údaje a vlastnosti/nastavení. Každý uživatel má, mimo jiné, možnost využít externí způsob přihlášení pomocí protokolu LDAP, pouze zadáním URL. Uživatel má dále definovaný jazyk, ve kterém komunikuje. Jazyk je představován entitou *Language* a navázán

na uživatele. Entita *UserRole* představuje uživatelské role, kterých může uživatel nabýt. Uživatel má díky tomu odpovídající práva a možnosti v nástroji. Předposlední entitou segmentu je *ProjectUser*, která tvoří spojení mezi uživatelem a projektem, do kterého je uživatel zařazen. Poslední entitou je *Company*, která slučuje uživatele do pomyslných skupin – společností, jejíž uživatelé spolu sdílí projekty, metriky a celkově informace a data.

Projektový segment je složen z pěti databázových entit. Hlavní entitou je entita *Project*, která obsahuje detailní informace o projektech. Na entitu *Project* je navázána entita *ProjectMetric*, která je vazební entitou mezi generickou metrikou a projektem, představující konkrétní metriky, využité v daných projektech. Dále entita *ProjectMetric* představuje jednotlivé nastavení pro získání dat metriky a nastavení upozornění uživatele na nestandardní hodnotu. Entita *ProjectMetricSnapshot* představuje historické informace („snímky“ dat) o načtení dat ze zdrojového rozhraní – nástroje IBM DNG. Entita *ProjectMetricColumnValue* představuje uložené historické hodnoty sloupců metriky pro dané „snímky“ dat. Poslední entitou segmentu je *ProjectMetricLog*, která představuje uložené zprávy týkající se projektové metriky, konkrétně například informace o průběhu a výsledku zpracování načítání dat z nástroje IBM DNG.

Segment metrik je největším ze všech segmentů, jelikož se skládá z šesti entit. Hlavní entitou je entita *Metric*, která představuje generické šablony metrik, vytvořené jak uživatelem (návaznost na uživatelovu firmu), tak i definované od počátku a nastavené jako veřejné, tudíž viditelné všem ostatním uživatelům nástroje. Každá metrika může být určitého obecného typu, takový typ definuje entita *MetricType*. Hodnota metriky se skládá/počítá z určitých sloupců, tyto sloupce definuje entita *MetricColumn*. Metrika může ovlivňovat určitou vlastnost/obor projektu, jakými jsou například kvalita a vysledovatelnost, takové vlastnosti představuje entita *AffecteField*. Jednou z důležitých vlastností metrik je jejich příslušnost k procesu ze standardu ASPICE, které představuje entita *AspiceProcess*. Poslední entitou je *AspiceVersion*, která obsahuje verze standardu ASPICE, které je potřeba ukládat pro případ změn stávajících či definování nových procesů v budoucích verzích ASPICE.

Čtvrtý a zároveň nejmenší segment správy aplikace, obsahuje pouze dvě entity. První entita *AppError* představuje zachycené chyby a chybové hlášení vytvořené za běhu webové aplikace a všech jejích navazujících součástí (vyjma IBM Jazz). Druhou entitou je *Setting*, která představuje různorodé nastavení aplikace, jako například nastavení emailové schránky nebo nastavení JSON Web Tokens (JWT).

### 6.4.2 Aplikační vrstva

Aplikační vrstva je tvořena webovým rozhraním REST API. REST API bylo zvoleno z důvodu, že se dnes jedná v podstatě o zažitý standard pro zvolenou architekturu podnikové aplikace. Toto webové rozhraní zajišťuje veškerou komunikaci mezi klientem (prezentační vrstvou) a databází (datovou vrstvou), to znamená, že zachytává veškeré požadavky uživatele, zpracovává je a odpovídá na ně. Aplikační vrstvu lze rozdělit do tří částí – první částí je samotná webová aplikace (REST API), druhá část je komponenta určená pro komunikaci s platformou IBM Jazz a třetí je komponenta pro připojení k databázi.

Komponenta pro komunikaci s platformou IBM Jazz – zdrojem dat pro metriky, se skládá z mikro servisu, který stahuje data z komponenty Jazz Reporting Service – uživatelem zadané URL adresy (viz podkapitola 6.1.2.8). Data se stahují pomocí protokolu HTTP a jeho metody GET. Po stažení jsou data ve formátu XML zpracována obecným analyzátorem, který zvládne zpracovat několik druhů formátů dat (dle všech získaných typů XML) a nabízí i flexibilní nastavení skrze proměnné parametry zadané uživatelem při vytváření předpisu metriky. Vyextrahované údaje z XML jsou poté uloženy do databáze a při požadavku zaslány na prezentační vrstvu, kde jsou zobrazeny uživateli ve formě grafu.

Komponentu pro připojení k databázi tvoří Entity Framework Core. Entity Framework Core (EF Core) je odlehčená, rozšiřitelná, open-source a platformě nezávislá verze frameworku pro přístup k datům. EF Core nabízí možnost objektově relačního mapování, což umožňuje vývojáři pracovat s databází pomocí objektů a eliminovat tak nutnost implementace datového přístupu k databázi. Entity Framework podporuje množství databázových systému, nejenom Microsoft SQL Server, ale i Oracle, MySQL, PostgreSQL, DB2, SQLite a další. Jednou z hlavních výhod EF Core je to, že data lze získávat z databáze pomocí metod/příkazů LINQ jazyka C#, které se svou syntaxí podobají SQL příkazům a tak přináší jednoduchý přístup k datům [49].

Samozřejmostí je také globální zachytávání chyb vzniklých za běhu programu. Tato funkcionalita je zajištěna pomocí přidané vrstvy (middleware). Middleware je software (třída nebo komponenta), který je zakomponován do pipeline aplikace pro zpracování požadavku a odpovědi. V tomto případě tedy třída obsahuje metodu, ve které se nachází pouze konstrukce try/catch spolu s voláním dalšího middleware. Konstrukce try/catch zachytí nastalou chybu v následných middleware třídách zařazených do pipeline. Takto zachycenou chybu uloží do databáze a navrátí odpověď s HTTP statusem 500 a chybovou zprávou v těle odpovědi.

**6.4.2.1 REST API** Roy Thomas Fielding [50] definuje Representional State Transfer (zkráceně REST) jako architektonický styl pro distribuované hypermediální systémy. Podstatou tohoto architektonického stylu je dodržení několika základních omezení. Prvním omezením je využití architektury klient-server, dalším omezením je, že komunikace musí být bezestavová, to znamená, že požadavek musí obsahovat všechny informace nutné k jeho porozumění. Další nutností je využití mezipaměti (cache) v případě, kdy o to požadavek explicitně požádá. Je kladen důraz

na jednotné rozhraní mezi komponentami – uniformní rozhraní. Nutností je také architektura systému složená z hierarchických vrstev. Posledním omezením je nepovinné Code-On-Demand.

REST využívá jako základní komunikační protokol HTTP a jeho operace, jakými jsou GET pro získání dat, POST pro vytvoření dat, PUT pro aktualizaci dat, DELETE pro smazání dat a PATCH pro částečnou aktualizaci dat. REST využívá jako primární datový formát pro požadavek a odpověď JSON, ale lze využít i jiné formáty, například XML. Ve výpisu 3 lze vidět ukázku požadavku zaslaného na REST API pomocí operace POST – požadavek na vytvoření dat.

```
POST /api/2.2/sites/9a8b7c6d-5e4f-3a2b-1c0d-9e8f7a6b5c4d/users HTTP/1.1
HOST: my-server
Content-Type: application/json

{
  "user": {
    "name": "NewUser1",
    "siteRole": "Publisher"
  }
}
```

Výpis 3: Ukázka požadavku na vytvoření dat zaslaného na REST API

**6.4.2.2 ASP.NET Core** ASP.NET Core je platformě nezávislý, výkonný, open-source framework pro tvorbu moderních internetových aplikací. Poslední verzí je verze 2.2 (verze 3.0 je v dubnu 2019 pouze jako preview), která je použita pro tvorbu všech webových aplikací vyvíjeného nástroje. Velkou výhodou ASP.NET Core je nezávislost na platformě, z čehož plyne, že je možné webovou aplikaci provozovat na jakémkoli operačním systému (Linux, MacOS nebo Windows), což je také hlavní výhodou oproti hojně využívané předchozí verzi ASP.NET 5.x. Platformní nezávislost je zajištěna tím, že je využito platformě nezávislého frameworku .NET Core, který je nástupcem .NET Framework. ASP.NET Core nabízí podporu pro vývoj a běh moderních RESTful webových aplikací. Mimo jiné také podporuje architekturu založenou na mikro servisech a propojení závislostí pomocí vkládání závislostí (dependency injection) [51].

```
[HttpPut("{id}")]
public async Task<BaseResponseModel> Put(int id, [FromBody]UserModel value)
{
   return await _userService.Edit(id, value);
}
```

Výpis 4: Ukázka controlleru ve frameworku ASP.NET Core psaná v jazyce C#

ASP.NET Core Web API vystavuje k přístupu jednotlivé metody controlleru, které přijímají data požadavku jako parametr a vracejí objekt (návratová data), která se zaobalí do odpovědi. Ve výpisu 4 lze vidět ukázku metody controlleru pro operaci PUT, v tomto případě aktualizaci údajů uživatele. Tato metoda se vyvolá zasláním požadavku PUT (například pomocí nástroje příkazového řádku curl nebo programu Postman) na adresu https://server.com/user/1, data požadavku se nacházejí v těle požadavku, jak je patrné z druhého parametru metody.

**6.4.2.3 Zabezpečení aplikační vrstvy** Aplikační vrstva je zabezpečena několika druhy zabezpečení. První z nich není přímým zabezpečením, jde spíš o doplněk k zabezpečení. Jde o kontrolu CORS v požadavku zaslaném na server. Použití CORS middleware umožňuje omezit příjem požadavků z jiných domén než explicitně specifikovaných. Dále lze omezit požadavky také dle druhu HTTP metody nebo dle existence požadované hlavičky v požadavku. Tato restrikce tedy zamezuje přístupu k datům serveru pro požadavky z jiné domény (například škodlivého webu), než na kterém je spuštěna instance prezentační vrstvy nástroje [52].

Dalším základním zabezpečením a dnes i standardem, vyžadovaným prohlížeči při přenosu citlivých dat, je použití protokolu HTTPS spolu s nezbytným SSL certifikátem. Takovéto zabezpečení je nutné jak pro aplikační vrstvu (REST API), tak i pro prezentační vrstvu (klientskou webovou aplikaci), jelikož přenášet nezašifrované informace po internetu mezi webovým prohlížečem a serverem je velmi nebezpečné a útočník by měl k těmto informacím volný přístup, což je v případě hesla nepřijatelné.

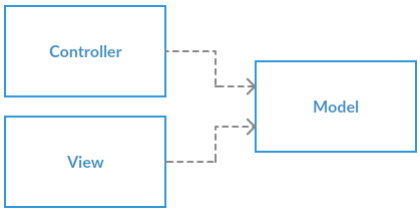
Zabezpečení přístupu k datům, neboli ověřování uživatele, probíhá pomocí technologie JSON Web Token (JWT). Při požadavku na přihlášení uživatele je uživatel autorizován v databázi dle uživatelského jména a hesla, poté je vygenerován unikátní JWT s určitou dobou expirace a uloženými základními uživatelskými údaji (ID a role uživatele). Uživatel má tedy po celou dobu, dokud token neexpiruje, možnost přistupovat k datům API. Poté, co token expiruje, je z API navrácena odpověď s HTTP statusem 401. Základní doba expirace tokenu je nastavena v databázi na 24 hodin. Uživatel je také autorizován k přístupu k datům dle přiřazené uživatelské role, kontrola role probíhá také na prezentační vrstvě a pro případ rozšíření řešení je přidána i kontrola na aplikační vrstvě, čili v REST API. Pokud uživatel nesplňuje příslušnost k dané roli, je navrácena odpověď s HTTP statusem 403 a požadavek není zpracován.

### 6.4.3 Prezentační vrstva

Prezentační vrstva vyvíjeného nástroje je tvořena webovou aplikací vytvořenou ve frameworku ASP.NET Core MVC. Webová aplikace je kompletně dostupná pouze registrovaným uživatelům, mimo části pro přihlášení a registraci. Hlavní část webové aplikace tvoří nástěnka projektu, kde se zobrazují veškeré metriky přiřazené administrátorem k projektu. Zde mohou uživatelé provádět operace s metrikami, jako například aktualizaci nebo export do dokumentů formátu docx, pptx nebo xlsx. Další části aplikace jsou primárně určeny uživatelům s administrátorskými právy, kteří v nich provádí nastavení metrik nebo vlastní společnosti a v případě super administrátora,

také celé aplikace. Konkrétní rozložení dostupné funkcionality dle uživatelských rolí je popsáno v podkapitole 6.3.

**6.4.3.1 Model-View-Controller** Architektonický návrhový vzor Model-View-Controller, často také zmiňovaný pod zkratkou MVC, dle definice Martina Fowlera [55] vytváří silné oddělení prezentace (View a Controller) od domény (Model), čili oddělení logiky od výstupu. Dále MVC dle Martina Fowlera rozděluje uživatelské rozhraní na Controller a View, kdy účelem Controlleru je reagovat na podněty uživatele a účelem View je zobrazovat stav Modelu za podmínky, že Controller a View spolu většinou nekomunikují přímo, ale přes Model. Na obrázku 22 lze vidět znázornění závislosti mezi Modelem, View a Controllerem. Martin Fowler záměrně vynechává návaznost mezi komponentami Controller a View, jelikož dle něj tato návaznost není využita.




Obrázek 22: Znázornění závislostí mezi Model, View a Controller dle Martina Fowlera [55]

Návrhový vzor MVC dle dokumentace ASP.NET Core MVC (autor Steve Smith) [53] rozděluje aplikaci na tři hlavní komponenty – Model, View a Controller. Dále zmiňuje, že při reálném aplikovaném použití tohoto vzoru v rámci ASP.NET Core MVC je uživatelský požadavek směrován na Controller, který je odpovědný za práci s Modelem za účelem provedení požadované akce a případného navrácení výsledku dotazu. Controller také vybírá View, které bude navráceno uživateli a do vybraného View případně dodá Model, který vyžaduje. Dodržení základních zásad architektonického vzoru umožňuje jednodušší vytváření, správu a testování zdrojového kódu vyvíjené webové aplikace.

**6.4.3.2 ASP.NET Core MVC** ASP.NET Core MVC je framework určený pro tvorbu webových aplikací využívajících architektonický návrhový vzor Model-View-Controller, zkráceně MVC. ASP.NET Core MVC je optimalizovaný pro použití s frameworkem ASP.NET Core, který je popsán v kapitole 6.4.2.2, kdy většina zmíněných výhod platí také pro ASP.NET Core MVC. ASP.NET Core MVC nabízí pokročilý způsob směrování, vkládání závislostí, filtrování požadavků, vysokou testovatelnost a mnoho dalšího [53].

Velkou výhodou je tvorba pohledů (View) pomocí Razor syntaxe. Razor je kompaktní, expresivní jazyk pro definování pohledů pomocí kódu napsaném v jazyce C#, Razor značek a HTML. Razor se používá k dynamickému generování webového obsahu na serveru, kdy nabízí možnost silně typového vázání modelu v pohledu. Generování na serveru lze brát jako výhodu i nevýhodu, ale v dnešní době, kdy jazyk JavaScript nabízí množství funkcí, lze pomocí něj jakoukoliv nevýhodu jazyka Razor doplnit. Díky možnosti využití programových konstrukcí jazyka C# je

možné využít množství klíčových slov jako například for, foreach, do, while, if, else, try/catch a mnoho dalších [54]. Ukázku pohledu vytvořeného pomocí jazyka Razor lze vidět ve výpisu 5. Ve výpisu je část typově vázaného pohledu, která po vygenerování na serveru zobrazí uživateli formulář se vstupním polem a popiskem.

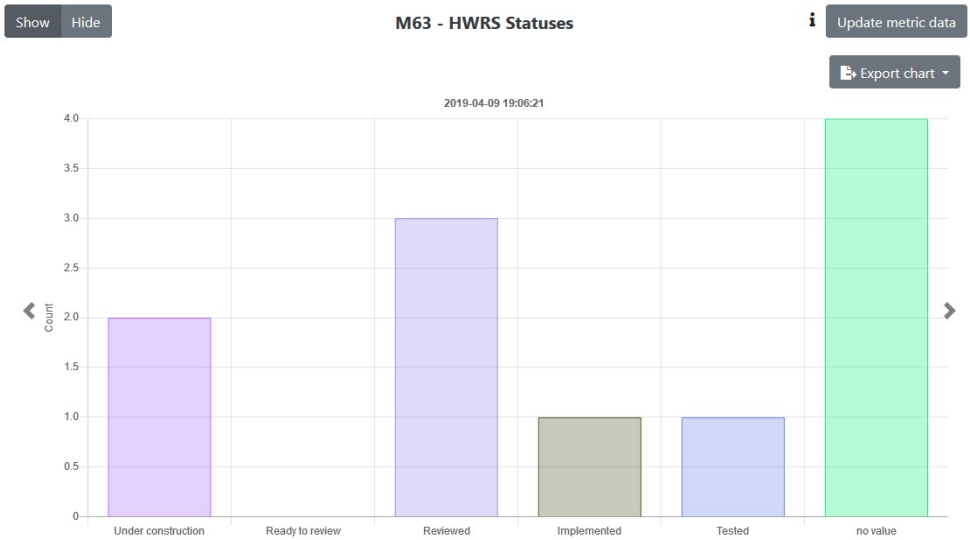
```
@model RegistrationViewModel
@{ ViewData["Title"] = "Registration"; }
<partial name="Partials/_ErrorTable" />
<form asp-controller="user" asp-action="registration" method="post" class="form
    -horizontal" role="form">
    <div class="col-md-10 mx-auto">
        <div class="form-row">
            <div class="form-group col-md-12">
                <label asp-for="Username" class="control-label"></label>
                <input asp-for="Username" class="form-control" placeholder="
                    Username" />
            </div>
        </div>
...
```

Výpis 5: Pohled vytvořený ve frameworku ASP.NET Core MVC

**6.4.3.3 Zabezpečení** Zabezpečení prezentační vrstvy je zajištěno pomocí přihlášení uživatele. Po přihlášení se uloží do prohlížeče uživatele cookie s dobou expirace nastavenou na dobu expirace přístupového tokenu (JWT) získaného z REST API – tato expirace platí v případě, že uživatel do té doby neukončí instanci internetového prohlížeče. V druhém případě má cookie nastavenou expiraci pro jednu session prohlížeče, to znamená, že pokud uživatel zavře prohlížeč, je také automaticky odhlášen – přístupová cookie je smazána. Pokud případně dojde k expiraci cookie (zároveň i přístupového tokenu), je uživatel automaticky odhlášen a přesměrován na přihlašovací stránku webové aplikace.

Samozřejmostí je tak, jako u webové aplikace aplikační vrstvy (REST API), použití protokolu HTTPS a SSL certifikátu. Toto zabezpečení je nezbytné, jelikož se z klientské stanice (uživatelova zařízení) přenáší přihlašovací jméno a heslo a v případě, že by nebyla komunikace šifrována, měl by k těmto údajům potencionální útočník kdykoli přístup a mohl by tyto údaje zneužít.

**6.4.3.4 Funkce nástěnky** Nástěnka projektu je pomyslným středobodem celé webové aplikace prezentační vrstvy. Dodává uživateli informace o stavu projektu skrze metriky vyobrazené formou grafu. Snímek metriky projektu přímo z části nástěnky projektu ve webové aplikaci lze vidět na obrázku 23.




Obrázek 23: Projektová metrika zobrazená na nástěnce projektu webové aplikace

Na obrázku 23 je vyobrazena metrika s názvem M63, zobrazující stav nebo lépe řečeno rozdělení stavů všech hardwarových požadavků. Bližší informace o metrice uživatel získá umístěním kurzoru myši na symbol informace v pravé horní části prostoru metriky. Jedná se o metriku množství, tudíž je zobrazena ve formě sloupcového grafu, kde jednotlivé sloupce představují stavy požadavku a výška sloupce počet výskytů stavu. Historický stav metriky lze zobrazit kliknutím na šipku vlevo od grafu. Pokud si uživatel přeje exportovat graf/y, má možnost provést export do různých formátů a to obrázek (png), dokument Microsoft Word (docx), pracovní sešit Microsoft Excel (xlsx) nebo prezentace Microsoft PowerPoint (pptx). V případě exportu grafů, dojde k exportu všech grafů – i všech historických, které byly načteny.

V případě, že projektová metrika využívá pro zobrazení spojnicového grafu, jedná se tedy o metriku pokrytí, lze v nástroji nastavit upozornění na dosažení minimální hodnoty metriky. To znamená, že v případě kdy hodnota metriky klesne pod uživatelem stanovou hodnotu, zobrazí se na nástěnce u patřičné metriky upozornění. Nástroj také nabízí možnost upozornění na klesající trend metriky, který indikuje, zda se hodnoty metriky stále nesnižují.

Dokumentaci k nástroji (webové aplikaci) a snímky z nástroje lze najít v příloze D.

**6.4.3.5 Využité knihovny** Prezentační vrstva využívá množství externích knihoven pro různorodé účely. Většina těchto knihoven je napsána v jazyce JavaScript. Knihovny v jazyce JavaScript nebyly vybrány náhodou, záměrem je moderní řešení dané problematiky, které sebou mimo jiné přináší benefit v podobě úspory objemu přenesených dat mezi klientským prostředím a serverem a také úsporu výkonu serveru, jelikož práce těchto knihoven probíhá v klientském prostředí – prohlížeči uživatele. Vybrané knihovny jsou s krátkým popisem vypsány v tabulce 7. Primárním aspektem při výběru knihoven bylo to, aby šlo o volně šiřitelné a bezplatné knihovny, ale také, aby nebylo nutné dělat kompromisy a ústupky v oblasti funkcionality nástroje.

| **Název** | **Popis** |
|---|---|
| docx | Knihovna docx je určená ke generování souborů ve formátu .docx z prostředí jazyků JavaScript a TypeScript. [56] |
| PptxGenJS | PptxGenJS je knihovna určená k vytváření PowerPoint prezentací, dostupná pro většinu prohlížečů. [57] |
| ExcelJS | ExcelJS je knihovna pro čtení, manipulaci a zápis tabulkových dat a stylů pro soubory XLSX a JSON. [58] |
| Chart.js | Chart.js je knihovna pro vykreslování grafů založená na HTML5, využívající tag *<canvas>*. [59] |

Tabulka 7: Vybrané knihovny využité ve webové aplikaci vyvíjeného nástroje

## 6.5 Servisní vrstva

Servisní vrstva je tvořena službou, která je spuštěna na pozadí operačního systému. Tato služba v pravidelných intervalech provádí aktualizaci hodnot metrik – stažení dat a následné vypočtení aktuálních hodnot všech projektových metrik v rámci celého nástroje. Pro tento účel využívá komponentu aplikační vrstvy pro komunikaci s platformou IBM Jazz, viz 6.4.2. Služba umožňuje nastavení intervalu spouštění přímo ve webové aplikaci, kde je tato možnost dostupná pouze uživateli v roli SuperAdmin.

Služba je vytvořena ve formě konzolové aplikace .NET Core, s přidanou podporou zpracování signálů daného operačního systému, jelikož takto vytvořenou aplikaci lze provozovat na libovolném operačním systému – Windows, Linux nebo MacOS. Služba zapisuje do konzolového bufferu operačního systému zprávy o svém stavu, čímž lze při zobrazení informací/statusu služby zjistit podrobnější stav služby a výsledky zpracování dat. Instalace služby, jak pro účely produkčního nasazení, tak i spuštění z vývojového prostředí, je popsána v příloze C.

## 6.6 Další použité technologie a nástroje

Mimo již dříve zmíněné technologie a nástroje je nutné zmínit i některé další nezbytné technologie a nástroje, které podporovaly a usnadňovaly vytvoření nástroje pro sledování a zobrazování metrik.

### 6.6.1 .NET Core

Open-source vývojová platforma .NET Core je vyvíjena a spravována společností Microsoft a .NET komunitou. Jedná se o odlehčeného nástupce frameworku .NET s nímž je zpětně kompatibilní. Hlavní předností oproti .NET Frameworku je nezávislost aplikace na specifické platformě, to znamená, že aplikace vytvořené pomocí platformy .NET Core lze provozovat na různých operačních systémech (Windows, MacOS nebo Linux). Výhodou je také vyšší rychlost oproti svému předchůdci nebo možnost škálovatelnosti.

.NET Core plně podporuje programovací jazyky C# a F#, částečně také VisualBasic. Poslední verzí je verze 2.2 a v polovině roku 2019 je plánována verze 3.0, nyní, v dubnu 2019, ve verzi preview. .NET Core podporuje čtyři platformě nezávislé scénáře aplikací – ASP.NET Core webové aplikace (využity ve vyvíjeném nástroji), aplikace běžící v příkazové řádce, knihovny a aplikace pro univerzální platformu Windows [60].

### 6.6.2 Konfigurační management

Jako repozitář pro zdrojový kód byl zvolen server `https://github.com/` [61]. Kde je dostupný kompletní, v maximálně míře dokumentovaný, zdrojový kód vyvíjeného nástroje, spolu s veškerými SQL skripty (definice tabulek i data – DML i DDL).

Repozitář obsahuje tři vývojové větve („branch“), hlavní vývojová větev *master* obsahuje spustitelnou verzi aplikace vyvíjenou v .NET Core. Další vývojová větev s názvem *react* obsahuje započatou aplikaci prezentační vrstvy vyvíjenou ve frameworku React. Třetí vývojová větev s názvem *netframework* obsahuje započatou aplikaci prezentační vrstvy vyvíjenou ve frameworku ASP.NET MVC verze 5. Jak webová aplikace React, tak i ASP.NET MVC byly zkoušeny pro porovnání se zvoleným řešením v ASP.NET Core, kdy bylo nakonec od těchto možnosti upuštěno a vývoj prezentační vrstvy pokračoval v ASP.NET Core. Obě vývojové větve, jak *netframework*, tak i *react*, mají návaznost na REST API, které je vyvíjeno pouze ve větvi *master*. SQL skripty jsou také vyvíjeny pouze ve větvi *master*.

Všechny zdrojové kódy se nacházejí také v příloze této práce (jedná se o rozšířenou kopii git repozitáře, vyjma vývojových větví *netframework* a *react*, které jsou pouze zkušební a určené pro případný další rozvoj nástroje).

## 6.7 Možnosti dalšího rozšíření nástroje

Vývoj nástroje je ve fázi, kdy je nástroj plně použitelný pro svůj daný účel. I přes to, že je funkcionalita nástroje naplněna dle zadání práce, nabízí se příležitosti, jak jej dále rozvinout či vylepšit. Jednou z hlavních možností rozšíření je implementace různých druhů napojení na zdroje dat, viz například podkapitola 6.1.2. Jedním ze zmíněných druhů napojení je například napojení na služby OSLC, které by umožnilo přístup k informacím z IBM DOORS Next Generation, potažmo i více nástrojů platformy IBM Jazz, v podobě, kdy by nebylo nutné prvotní nastavení daného zdrojového nástroje – celé nastavení by probíhalo pouze při definici metriky uživatelem v nástroji pro sledování metrik.

Dalším benefitem implementace OSLC, či jiného přístupu, by byla teoretická možnost rozšíření napojení i na jiné nástroje z oblasti správy požadavků než je IBM DOORS Next Generation. Teoretickou možností je i samotné rozšíření napojení na jiné nástroje (i jinak než skrze služby OSLC) jakými jsou například Enterprise Architect nebo codeBeamer, které jsou v praxi také hojně využívány.

Rozšířit nástroj lze také přidáním podpory více druhů metrik a to jak z pohledu výsledných grafů, tak i z pohledu jiných typů metrik, které se netýkají primárně správy požadavků. Například metriky týkající se požadavků na změny zdrojového kódu, chybové hlášení atp., jelikož tuto agendu vývojové týmy také hojně zpracovávají a je vhodné mít možnost přehledně zobrazit stav takovéto agendy pomocí metrik.

# 7 Závěr

Tuto práci lze rozdělit do dvou částí, první část je převážně teoretická, týkající se softwarového vývoje, měření kvality softwarového vývoje a z toho plynoucích metrik. První část je také z části průzkumná, kdy jsou zkoumány dostupné řešení z oblasti správy požadavků a měření kvality – metrik. Druhá část práce se již plně zabývá hlavním předmětem této diplomové práce, to je vytvořením prototypového nástroje pro sledování a zobrazování metrik.

Standardy pomáhají řídit a zlepšovat procesy softwarového vývoje. Jedním z takových standardů je Automotive SPICE. Dodržování standardu sebou přináší mnoho výhod, ať už jde o efektivnější vývoj a správu softwaru, tak i v neposlední řadě dnes často opomíjenou kvalitu výsledného softwaru. Posouzení kvality prováděných procesů a celkové procesní vyspělosti lze hodnotit. Takovéto hodnocení se vytváří na základě přímých i nepřímých výsledků procesu a mnoha dalších aspektů. V této chvíli přichází na řadu metriky. Metriky tvoří nezbytnou část profesionálního vývoje softwaru, jelikož srozumitelně reflektují například kvalitu softwaru i celého projektu. Metriky tvoří výstup, který při správné reprezentaci lze využít pro optimalizaci procesů. Jedním z hlavních způsobů reprezentace metriky je číselná hodnota/y, která se poté prezentuje například ve vhodné formě grafu.

Metriky lze využít v rámci standardu Automotive SPICE ke sledování procesů mnoha domén. V kontextu této práce byly sledovány metriky z domény správy požadavků a to nejen softwarových, ale i například hardwarových. Metriky se váží k různým procesům standardu Automotive SPICE a pomáhají popisovat, v jakém stavu se nacházel a nachází daný sledovaný proces.

Všechny požadavky je vhodné zpracovávat a uchovávat v přehledné formě, k tomu slouží množství dnes nabízených nástrojů. Jedním z předních je platforma IBM Jazz a její nástroj IBM DOORS Next Generation. V tomto nástroji lze pohodlně spravovat požadavky a veškeré návaznosti na ně. Problémem tohoto nástroje je, že nenabízí možnost vytváření, ani generického zobrazování, jakýchkoli metrik. Tudíž bylo nutné najít jiný nástroj, který by tuto funkcionalitu ve spolupráci s IBM Jazz poskytoval. Bylo proto zkoumáno několik nástrojů, ale žádný z nich neposkytoval takovou funkcionalitu, která by splňovala zadání na vytváření a zobrazování metrik. Některé nástroje vykazovaly náznaky možnosti práce s metrikami, ale bohužel vždy v nedostatečné formě, kdy nebylo možné flexibilně nastavit parametry, zobrazení aj. Proto bylo tedy přistoupeno k vytvoření vlastního řešení, které bylo vyvíjeno od počátku, s návazností na nástroj IBM DOORS Next Generation, potažmo platformu IBM Jazz.

Cílem vyvíjeného nástroje bylo to, aby umožňoval sledování a zobrazování metrik, možnost definice uživatelských metrik a komplexní správu projektu i celé aplikace. Nutnou prerekvizitou pro vývoj samotného nástroje je dostupná instance IBM Jazz, která slouží jako zdroj pro testovací data a také jako učební pomůcka pro bližší porozumění celé platformě IBM Jazz a to nejen pro účely této práce, ale také pro potřeby dalších studentů. Po náročné instalaci byla platforma zprovozněna a naplněna daty – požadavky. Poté bylo nutné se na platformu napojit a získat v ní uložená data. Bylo zjištěno množství způsobů, jak proces exportu dat provádět. Jednotlivá

řešení byla prozkoumána a zhodnocena, kdy nakonec byla vyhodnocena čtyři použitelná řešení, z nichž jedno bylo po konzultaci s vedoucím práce vybráno – jmenovitě Jazz Reporting Service.

Jazz Reporting Service nabízí nejpřívětivější možnost exportu dat pomocí webového rozhraní a formátu XML, kdy se obejde plně bez účasti uživatele (výjimkou je pouze počáteční nastavení exportu). Hlavní výhodou exportu je jeho automatické spouštění, které vyžaduje pouze zmíněné počáteční nastavení v nástroji IBM DOORS Next Generation a vyvíjeném nástroji. Nicméně služby OSLC poskytují do budoucna možnost rozšíření nebo nahrazení stávajícího řešení exportu dat pomocí Jazz Reporting Service, díky možnosti napojení na vícero nástrojů z domény správy požadavků, nejen IBM DOORS Next Generation.

Vyvinutý prototypový nástroj je koncipován jako podniková webová aplikace. Takováto webová aplikace se skládá ze serverové části a klientské části. Uživatel má tedy přístup pouze ke klientské části, se kterou má možnost skrze internetový prohlížeč dále pracovat. Všechny funkce nástroje jsou přístupné pouze registrovaným uživatelům, aby bylo zajištěno maximální bezpečí citlivých firemních dat a informací. Nástroj umožňuje uživateli nastavení libovolné metriky i flexibilní možnost načtení informací ze zdroje dat pomocí proměnných parametrů. Metriky (hodnoty metrik) jsou v nastavitelném intervalu aktualizovány, případně je uživatel může v nástroji aktualizovat manuálně. Díky tomu je možné sledovat aktuální stav procesů/projektu přehledně ve formě grafů na nástěnce projektu. Důležitou funkcionalitou je také možnost exportu výsledných grafů do různých formátů, ať už pro potřeby revize (například obrázek nebo dokument Microsoft Word) nebo prezentace zákazníkovi (prezentace Microsoft PowerPoint). Uživatelé jsou členěni do rolí, kdy každá z rolí má určitou odpovědnost – odpovídající možnost správy nástroje. Uživatel v roli SuperAdmin má tedy možnost kompletní správy aplikace s možností konfigurace všech parametrů. Další uživatelské role (User a Admin) mají klasické uživatelské možnosti, plynoucí z názvu jejich rolí.

Nástroj je tedy plně k dispozici pro řešení problematiky sledování a zobrazování metrik a usnadnění administrace projektu podléhajícího standardu Automotice SPICE. Jak již bylo krátce zmíněno, nástroj je také možné dále rozšířit, ať už z pohledu způsobu získávání dat (služby OSLC) nebo také napojení na jiné nástroje z oblasti správy požadavků.

# Literatura


[1] SOMMERVILLE, Ian. Software engineering. 9th edition, International Computer Science Series, 2011. Kapitola 1.1, s. 9-10. ISBN: 978-0-13-703515-1

[2] SOMMERVILLE, Ian. Software engineering. 9th edition, International Computer Science Series, 2011. Kapitola 26, s. 706-711. ISBN: 978-0-13-703515-1

[3] PRESSMAN, Roger S. Software engineering: a Practioner's Approach. Seventh edition, 2010. Kapitola 2.2, s. 37-38. ISBN 978–0–07–337597–7

[4] Automotive SPICE Process Reference and Assessment Model. VDA QMC [online]. Release 3.1, 1. listopadu 2017 [cit. 2018-12-22].
Dostupné z: `http://www.automotivespice.com/download/`

[5] Automotive SPICE Process Reference and Assessment Model. VDA QMC [online]. Release 3.1, 1. listopadu 2017. Kapitola 3, s. 11 [cit. 2018-12-22].
Dostupné z: `http://www.automotivespice.com/download/`

[6] Automotive SPICE Process Reference and Assessment Model. VDA QMC [online]. Release 3.1, 1. listopadu 2017. Kapitola 3.3.3, s. 23 [cit. 2018-12-22].
Dostupné z: `http://www.automotivespice.com/download/`

[7] SOMMERVILLE, Ian. Software engineering. 9th edition, International Computer Science Series, 2011. Kapitola 24.4, s. 668-677. ISBN: 978-0-13-703515-1

[8] SOMMERVILLE, Ian. Software engineering. 9th edition, International Computer Science Series, 2011. Obrázek 24.9, s. 669. ISBN: 978-0-13-703515-1

[9] Automotive SPICE v3.1 Pocket Guide. Kugler Maag CIE [online]. květen 2018, s. 20-25 [cit. 2018-02-15]. ISBN 978-3-946547-21-2
Dostupné z: `https://www.kuglermaag.de/fileadmin/05_CONTENT_PDF/2-10_automotive-spice_version_3_pocketguide.pdf`

[10] Automotive SPICE Process Reference and Assessment Model. VDA QMC [online]. Release 3.1, 1. listopadu 2017, Kapitola 4.6.3, s. 72-73 [cit. 2018-12-26].
Dostupné z: `http://www.automotivespice.com/download/`

[11] Famous Inventions to Come from IBM. IBM - Big Blue [online]. [cit. 2019-02-03].
Dostupné z: `http://www.aboutbigblue.com/famous-inventions-to-come-from-ibm/`

[12] Produkty a služby společnosti IBM. IBM [online]. [cit. 2019-02-03].
Dostupné z: `https://www.ibm.com/cz-cs/products`

[13] Jazz. IBM [online]. [cit. 2019-02-03].
Dostupné z: https://jazz.net/story/about/

[14] Linked Data. W3C [online]. [cit. 2019-02-03].
Dostupné z: https://www.w3.org/standards/semanticweb/data

[15] IBM Rational Lifecycle Integration Adapters - Tasktop. IBM [online]. [cit. 2019-02-05].
Dostupné z: https://jazz.net/products/rational-adapters-tasktop-edition

[16] Jazz Products. IBM [online]. [cit. 2019-02-03].
Dostupné z: https://jazz.net/products

[17] IBM Rational DOORS Next Generation. IBM [online]. [cit. 2019-02-03].
Dostupné z: https://jazz.net/products/rational-doors-next-generation/

[18] Metrics Dashboard for DOORS. River North Solutions, INC. [online]. [cit. 2019-01-14].
Dostupné z: http://rivernorthsolutions.com/Requirement_Metrics_Dashboard.pdf

[19] codeBeamer ALM. Intland Software [online]. [cit. 2019-01-15].
Dostupné z: https://intland.com/application-lifecycle-management/

[20] Enterprise Architect. Sparks Systems [online]. [cit. 2019-01-17].
Dostupné z: https://sparxsystems.com/products/ea/14/index.html

[21] BELL, Aaron. MDG Link for DOORS User Guide [online]. Sparx Systems Pty Ltd
[cit. 2019-01-18]. Dostupné z: https://sparxsystems.com/downloads/pdf/DOORS.pdf

[22] SpiraTeam. Inflectra Corporation [online]. [cit. 2019-01-27].
Dostupné z: https://www.inflectra.com/SpiraTeam/

[23] ReqTest. ReqTest AB [online]. [cit. 2019-01-29].
Dostupné z: https://reqtest.com/

[24] Process Street [online]. [cit. 2019-01-19].
Dostupné z: https://www.process.st/product/

[25] Confluence, Atlassian [online]. [cit. 2019-01-20].
Dostupné z: https://www.atlassian.com/software/confluence

[26] JIRA Core, Atlassian [online]. [cit. 2019-01-20].
Dostupné z: https://www.atlassian.com/software/jira/core

[27] Easy projects [online]. [cit. 2019-01-28].
Dostupné z: https://www.easyprojects.net/project-management-tools/

[28] Software as a service. Microsoft [online]. [cit. 2019-01-28].
Dostupné z: https://azure.microsoft.com/cs-cz/overview/what-is-saas/

[29] System Requirements for IBM Rational Collaborative Lifecycle Management (CLM) 6.0.6. Susan Yeshin, IBM [online]. [cit. 2019-04-01].
Dostupné z: https://jazz.net/wiki/bin/view/Deployment/CLMSystemRequirements606

[30] Rational Collaborative Lifecycle Management Solution 6.0.6, detailed system information. IBM [online]. [cit. 2019-04-01].
Dostupné z: https://www.ibm.com/software/reports/compatibility/clarity-reports/report/html/softwareReqsForProduct?deliverableId=4167CD00DF5711E7974C181B76870538&osPlatforms=Linux&duComponentIds=D004|D006|D001|D002|S003|S005&mandatoryCapIds=30|9|24|35|13|132|42|19|16|26|40&optionalCapIds=133|66|135|7|5|12|1|242|187|136|19|137|27|4&cm_mc_uid=21724745906315384077711&cm_mc_sid_50200000=65120341554383588921

[31] CLM Sizing Strategy for the 6.x releases. Vaughn Rokosz, Steven Beard, IBM [online]. [cit. 2019-04-01].
Dostupné z: https://jazz.net/wiki/bin/view/Deployment/CLMSizingStrategy60

[32] IBM Rational Team Concert download. IBM [online]. [cit. 2019-04-01].
Dostupné z: https://jazz.net/downloads/rational-team-concert

[33] About WebSphere Liberty. IBM [online]. [cit. 2019-04-01].
Dostupné z: https://developer.ibm.com/wasdev/websphere-liberty/

[34] IBM Db2. IBM [online]. [cit. 2019-04-07].
Dostupné z: https://www.ibm.com/analytics/cz/cs/technology/db2/

[35] VS Client API Usage. IBM [online]. [cit. 2019-04-08].
Dostupné z: https://jazz.net/wiki/bin/view/Main/APIUsage

[36] Reportable REST API. IBM [online]. [cit. 2019-04-08].
Dostupné z: https://jazz.net/wiki/bin/view/Main/ReportsRESTAPI

[37] DNG Reportable REST API. IBM [online]. [cit. 2019-04-08].
Dostupné z: https://jazz.net/wiki/bin/view/Main/DNGReportableRestAPI

[38] Client extension API for the Requirements Management (RM) application. IBM [online]. [cit. 2019-04-08].
Dostupné z: https://jazz.net/wiki/bin/view/Main/RMExtensionsAPI605

[39] Requirements Interchange Format (ReqIF). Object Management Group [online]. Verze 1.2, červenec 2016 [cit. 2019-04-08].
Dostupné z: https://www.omg.org/spec/ReqIF/1.2/PDF

[40] Exporting artifacts from a requirements project into ReqIF files.
IBM [online]. [cit. 2019-04-08].
Dostupné z: https://www.ibm.com/support/knowledgecenter/SSYMRC_6.0.3/com.ibm.rational.rrm.help.doc/topics/t_export_reqif.html

[41] Rational DOORS Next Generation ReqIF API 1.0. IBM [online]. [cit. 2019-04-08].
Dostupné z: https://jazz.net/wiki/bin/view/Main/DNGReqIF

[42] OSLC. OASIS [online]. [cit. 2019-04-08].
Dostupné z: https://open-services.net/

[43] OSLC Requirements Management Version 2.1. Part 1: Specification.
OASIS Open [online]. 24. srpna 2018 [cit. 2019-04-08].
Dostupné z: http://docs.oasis-open.org/oslc-domains/oslc-rm/v2.1/cs01/part1-requirements-management-spec/oslc-rm-v2.1-cs01-part1-requirements-management-spec.pdf

[44] DNG Module API Overview. IBM [online]. [cit. 2019-04-08].
Dostupné z: https://jazz.net/wiki/bin/view/Main/DNGModuleApiOverview

[45] Introduction to the Jazz Reporting Service. IBM [online]. [cit. 2019-04-08].
Dostupné z: https://www.ibm.com/support/knowledgecenter/SSYMRC_6.0.6/com.ibm.team.jp.jrs.doc/topics/c_report_builder_ov.html

[46] Choosing the right data source. IBM [online]. [cit. 2019-04-08].
Dostupné z: https://www.ibm.com/support/knowledgecenter/SSYMRC_6.0.6/com.ibm.team.jp.jrs.doc/topics/c_jrs_choose_ds.html#c_jrs_choose_ds

[47] SQL Server 2017. Microsoft [online]. [cit. 2019-04-10].
Dostupné z: https://www.microsoft.com/en-us/sql-server/sql-server-2017

[48] SQL Server Database Engine Backward Compatibility. Microsoft [online]. [cit. 2019-04-10].
Dostupné z: https://docs.microsoft.com/en-us/sql/database-engine/sql-server-database-engine-backward-compatibility?view=sql-server-2017

[49] Entity Framework Core. Microsoft [online]. [cit. 2019-04-10].
Dostupné z: https://docs.microsoft.com/cs-cz/ef/core/

[50] Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000. [cit. 2019-04-10].
Dostupné z: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[51] Introduction to ASP.NET Core. Microsoft [online]. [cit. 2019-04-10].
Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/core/?view=aspnetcore-2.2

[52] Enable Cross-Origin Requests in ASP.NET Core. Microsoft [online]. [cit. 2019-04-10].
Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/core/security/cors?view=aspnetcore-2.2

[53] Overview of ASP.NET Core MVC. Microsoft [online]. [cit. 2019-04-11].
Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/core/mvc/overview?view=aspnetcore-2.2

[54] Razor syntax reference for ASP.NET Core. Microsoft [online]. [cit. 2019-04-11].
Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/core/mvc/views/razor?view=aspnetcore-2.2

[55] FOWLER, Martin. GUI Architectures [online]. [cit. 2019-04-11].
Dostupné z: https://www.martinfowler.com/eaaDev/uiArchs.html

[56] DOCX [online]. [cit. 2019-04-11].
Dostupné z: https://docx.js.org/

[57] PptxGenJS [online]. [cit. 2019-04-11].
Dostupné z: https://gitbrent.github.io/PptxGenJS/docs/quick-start.html

[58] ExcelJS [online]. [cit. 2019-04-11].
Dostupné z: https://github.com/exceljs/exceljs

[59] Chart.js [online]. [cit. 2019-04-11].
Dostupné z: https://www.chartjs.org/

[60] LANDER, Rich. About .NET Core. Microsoft [online]. [cit. 2019-04-11].
Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/core/about

[61] Jazz Metrics. GitHub [online]. [cit. 2019-04-07].
Dostupné z: https://github.com/michalprikryl/jazz-metrics

# A Seznam příloh

- /
  - InstallationJazz
    - limits.conf ..... Soubor s upraveným nastavením limitů OS
  - JazzMetrics ..... Kompletní zdrojové kódy nástroje
    - Database ..... Přístup k databázi – ORM
    - JazzMetrics ..... Testovací konzolová aplikace
    - Library ..... Knihovna nástroje
    - Service ..... Služba pro aktualizaci metrik
    - WebAPI ..... Aplikační vrstva – REST API
    - WebApp ..... Prezentační vrstva
    - JazzMetrics.sln ..... Visual studio řešení (solution)
    - JazzMetrics.pdf ..... Informace o dostupnosti nástroje
  - Service
    - jazz-metrics.service ..... Konfigurační soubor služby
  - SQL
    - database_clean.sql ..... Skript pro smazání databáze
    - database_ddl.sql ..... Skript pro vytvoření databáze
    - database_diagram.png ..... ER diagram databáze
    - database_dml.sql ..... Skript pro naplnění databáze
    - JazzMetrics.ndm ..... Model databáze pro program Navicat

# B Návod k instalaci IBM Jazz

Instalace platformy IBM Jazz vyžaduje znalost práce v prostředí operačního systému Linux a také uživatelská oprávnění na úrovni root. Návod na instalaci byl sestavován nad operačním systémem Ubuntu verze 18.04.01 LTS Bionic Beaver. Kompletní návod lze najít na následující URL adrese, kde se nachází formulář, který na základě uživatelem vyplněných parametrů sestaví návod k instalaci platformy IBM Jazz. Následující návod je rozšířenější a je doplněn o poznámky autora. Další praktické informace týkající se instalace IBM Jazz lze nalézt v kapitole 6.1.

1. Stažení Collaborative Lifecycle Management 6.0.6 (IBM Jazz) z následující URL adresy.

2. Dalším krokem je rozbalení stažené komprimované složky pomocí příkazu `unzip CLM-Web-Installer-Linux-6.0.6.zip -d jazz`, čímž se obsah rozbalí do složky *jazz*.

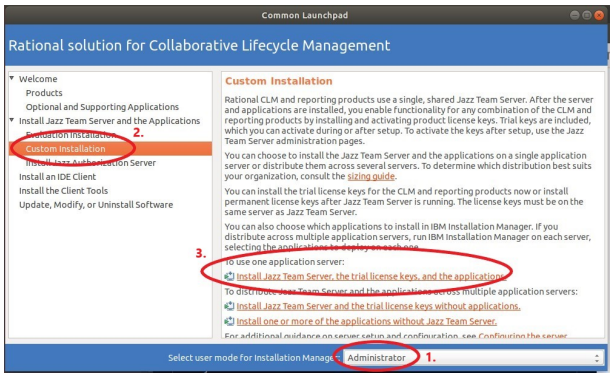
3. Před samotnou instalací je žádoucí navýšit limity počtu otevřených souborů a uživatelských procesů. To se provede úpravou souboru */etc/security/limits.conf*. Soubor lze otevřít pomocí příkazu `sudo pico /etc/security/limits.conf` a přidat nastavení z výpisu 6 (soubor se také nachází v příloze *InstallationJazz/limits.conf*).

```
* hard nofile  65536
* soft nofile  65536
* hard nproc 10000
* soft nproc 10000
```

Výpis 6: Přidaný obsah souboru /etc/security/limits.conf

4. Po modifikaci souborů je nutné restartovat operační systém.

5. Po restartu systému spustíme instalačního průvodce – `sudo ./ibm_jazz/launchpad.sh` a po pár sekundách se zobrazí rozhraní instalace. Hned po spuštění uživatelského rozhraní instalace je nutné vybrat v jeho spodní části uživatelský mód instalace **Administrator**.




Obrázek 24: Spuštění instalace IBM Jazz.

6. Samotná instalace se spouští výběrem možnosti menu *Custom Installation* a poté výběrem *Install Jazz Team Server, the trial licence keys and the application* – postup vyznačen na obrázku 24.

7. Pro spuštění instalace vyplníme uživatelské jméno a heslo svého účtu zřízeného na serveru `https://jazz.net/`, který je nutný také pro stažení platformy IBM Jazz a přístup k nápovědě a manuálům IBM Jazz.

8. Nyní následuje výběr nástrojů, které si přejeme nainstalovat. Pro účely správy požadavků a návazných domén je záhodno vybrat – Change and Configuration Management, IBM Jazz Reporting Service, Jazz Team Server, Requirement Management a Trial keys Collaborative Lifecycle Management products. Samozřejmě je možné vybrat instalované nástroje dle libosti, ale minimálním nástrojem, nutným pro provoz, je Jazz Team Server, který představuje základní kámen pro ostatní nástroje. Po výběru nástrojů je možné kliknout na tlačítko *Next* a pokračovat tak dále.

9. V některých případech je zobrazeno upozornění, že Váš operační systém nebyl rozpoznán. Pokud se toto upozornění zobrazí a jste si jistí, že aktuální operační systém je platformou IBM Jazz podporovaný, je možné přejít dále. Stejně tak i v případě jiných upozornění upozornění.

10. Nyní jsou zobrazeny licenční podmínky, se kterými je nutné pro přechod dále souhlasit.

11. Výběr destinace instalovaných aplikací ponecháme v počátečním nastavení a pokračujeme dále.

12. Stejně tak ponecháme nastavení destinace skupiny balíčků.

13. Instalované jazyky vybereme dle libosti (čeština není doporučena, z důvodu nepřesných a matoucích překladů).

14. Potvrdíme instalované nástroje/balíčky a pokračujeme kliknutím na tlačítko *Next*.

15. Nyní znovu ponecháme veškeré nastavení a klikneme na tlačítko *Next*. V případě, že je již na daném serveru nainstalován webový server WebSphere, nastavíme jej aplikováním (instalací) WAR souborů do existující složky.

16. Ponecháme základní nastavení i dalších dvou funkcionalit.

17. Ve shrnutí zkontrolujeme instalované nástroje a klikneme na tlačítko *Install*, čímž se spustí instalace. Instalace trvá určitou dobu v závislosti na rychlosti připojení k internetu. Instalace v této konfiguraci potřebuje přibližně okolo 4 GB místa na disku.

18. Po dokončení instalace klikneme na tlačítko *Finish*, čímž je instalace dokončena.

19. Nyní následuje instalace databáze. Pokud již máte databázi nainstalovanou nebo využíváte externí databázi DB2, je možné přejít na bod 27. Doporučená databáze je DB2 od společnosti IBM, jelikož obsažená databáze Apache Derby je nedostačující – umožňuje využití pouze 10 uživatelů.

20. Databázi je nutné nainstalovat v takové verzi, aby byla podporována – výčet takovýchto verzí se nachází v tabulce 5. Pro účely ukázkové instalace bylo využito IBM DB2 Express 11.1, ale postup bude shodný i pro jiné edice a verze IBM DB2. IBM DB2 Express lze stáhnout zde (je nutné vlastnit bezplatný účet IBMid).

21. Stažený soubor lze rozbalit pomocí příkazu `tar zxf v11.1_linuxx64_expc.tar.gz`. Před samotnou instalací DB2 je nutné nainstalovat určité balíčky, které vyžaduje DB2 jako prerekvizitu. Před instalací balíčku je žádoucí provést aktualizaci již nainstalovaných balíčku – `sudo apt update`. Instalace balíčku se provede příkazem `sudo apt install libx32stdc++6 libpam0g:i386 libaio1`.

22. Po úspěšné instalaci prerekvizit je možné spustit samotnou instalaci databázového serveru DB2, tu spustíme pomocí příkazu `sudo expc/db2setup` (spouštěný soubor se nachází v rozbalené složce).

23. Po spuštění okna instalace klikneme na tlačítko *Install New* a vybereme verzi – na výběr je pouze jedna.

24. Všechna nastavení instalace ponecháme v základním nastavení, pouze zatrhneme, že souhlasíme s IBM podmínkami užití a klikneme na tlačítko *Next*.

25. Databázový administrátorský účet (db2inst1) ponecháme stejný, pouze zadáme heslo, které je nutné si poznamenat pro další použití – například administraci databáze. Stejně tak i pro následujícího uživatele (db2fenc1).

26. Ve shrnutí instalace není nutné nic měnit, pouze spustit instalaci kliknutím na tlačítko *Install*.

27. Nyní následuje vytvoření databází potřebných pro provozování všech nainstalovaných nástrojů IBM Jazz. Potřebné příkazy jsou zadávány do příkazového řádku DB2, zároveň je nutné tyto příkazy provádět pomocí databázového účtu **db2inst1**. Přepnutí uživatele v terminálu se docílí tak, že v nové instanci terminálu zadáme příkaz `su - db2inst1` a zadáme heslo uživatele. Příkazová řádka DB2 se spustí zadáním příkazu `db2`. Veškeré příkazy pro tvorbu a úpravu databáze jsou uvedeny ve výpisu 7. Příkazy je nutné provést po jednom, ve stejném pořadí, v jakém jsou uvedeny.

```
create database JTS using codeset UTF-8 territory en PAGESIZE 16384
create database CCM using codeset UTF-8 territory en PAGESIZE 16384
create database RM using codeset UTF-8 territory en PAGESIZE 16384
create database DCC using codeset UTF-8 territory en PAGESIZE 16384
create database LQE using codeset UTF-8 territory en PAGESIZE 32768
update db cfg for LQE using maxappls 300
update db cfg FOR LQE using locklist 20000
update db cfg for LQE using LOGFILSIZ 20000
update db cfg for LQE using logprimary 25
update db cfg for LQE using logsecond 100
create database DW using codeset UTF-8 territory en PAGESIZE 32768
update db cfg for DW using maxappls 300
update db cfg FOR DW using locklist 20000
update db cfg for DW using LOGFILSIZ 20000
update db cfg for DW using logprimary 50
update db cfg for DW using logsecond 200
```

Výpis 7: Příkazy pro vytvoření a úpravu DB2 databází

28. Po instalaci je možné spustit WebSphere Liberty server. WebSphere server se spouští příkazem `sudo /opt/IBM/JazzTeamServer/server/server.startup`.

29. Pokud je žádoucí změna portu, na kterém instance IBM Jazz běží, je tuto změnu **nutné** provést nyní, jelikož po nastavení všech nástrojů je provedení takovéto změny velmi složité. Změna portu se provádí při vypnutém WebSphere serveru, toho se docílí spuštěním příkazu `sudo /opt/IBM/JazzTeamServer/server/server.shutdown`. Poté je nutné provést změnu portu v souboru *server.xml* a to pomocí příkazu `sudo pico /opt/IBM/JazzTeamServer/server/liberty/servers/clm/server.xml`. Změna se provede úpravou vlastnosti **httpPort**, případně **httpsPort**, elementu XML **httpEndpoint**. Defaultní hodnoty jsou 9080 a 9443, vhodné je tedy provést změnu na 80 a 443. Uložením souboru se změna aplikuje, poté je doporučeno restartovat server a až poté znovu spustit WebSphere server, stejně tak i databázový server, kdy se přihlásíme jako uživatel **db2inst1** a zadáme příkaz `db2start` (v některých případech je nutné spuštění provést po každém restartu).

30. Nyní nastává chvíle nastavení nástrojů. V prohlížeči zajdeme adresu `https://127.0.0.1/jts/setup` (případně vlastní adresu serveru), kde je nutné zadat přihlašovací údaje, ty jsou pro tento účel nastaveny – uživatelské jméno ADMIN a heslo ADMIN.

31. Na uvítací stránce vybereme *Custom setup* a přejdeme na dále.

32. Na stránce *Introduction* pouze klikneme na tlačítko *Next*.

33. Na stránce *Configure Public URI* ponecháme nastavení a klikneme na tlačítko *Next*.

34. Na stránce *Configure Database* vybereme v poli *Database Vendor* možnost **DB2**. Poté zadáme do pole *JDBC Location* hodnotu připojovacího řetězce databáze JTS (možno překopírovat z nápovědy) *//localhost:50000/JTS:user=db2inst1;password=password;* a do pole *JDBC Password* heslo uživatele db2inst1. Poté stiskneme tlačítko *Test Connection*, po testu připojení bude nutné vytvořit databázové tabulky, to lze provést kliknutím na nově zobrazené tlačítko *Create Tables*. Po vytvoření tabulek lze pokračovat na další stránku.

35. Na stránce *Configure E-mail Notification* vybereme pro *E-Mail Notification* možnost **disabled**, klikneme na tlačítko *Test Connection* a poté přejdeme na další stránku.

36. Na stránce *Register Applications* probíhá registrace nainstalovaných nástrojů. Pro přidání nástroje vložíme jeho URL do pole *Discovery URL*, potřebné URL adresy jsou uvedeny ve výpisu 8. Ostatní parametry se vyplní automaticky. Po vyplnění všech nástrojů (URL adres) klikneme na tlačítko *Register Applications* a po dokončení registrace nástrojů pokračujeme dále.

```
https://127.0.0.1/ccm/scr
https://127.0.0.1/dcc/scr
https://127.0.0.1/lqe/scr
https://127.0.0.1/rm/scr
https://127.0.0.1/rs/scr
```

Výpis 8: URL adresy jednotlivých nástrojů pro potřeby registrace

37. Na stránce *Setup User Registry* vybereme v kroku 1 možnost *Liberty Basic Registry*. V kroku 2 zadáme uživatelské jméno a heslo nového administrátorského uživatele, který nahradí uživatele ADMIN. V kroku 3 se přiřazují licence. Licenci je nutné zakoupit nebo využít licenci s omezenou platností, dodávanou s instalací.

38. Na stránce *Configure Data Warehouse* konfigurujeme databázi datového skladu. Konfigurace probíhá stejně jako v bodě 34, jen s tím rozdílem, že místo identifikátoru databáze JTS vkládáme identifikátor databáze DW. Ostatní pole se nechávají prázdné. Po otestování spojení přejdeme dále.

39. Nyní následuje nastavení jednotlivých nástrojů. Na stránce *Configure Database* probíhá konfigurace znovu stejně jako v bodě 34, s tím rozdílem, že místo identifikátoru databáze JTS vkládáme identifikátor dané databáze (jednotlivé nástroje se tedy konfigurují stejně, s jediným rozdílem, že se vyplňuje jiný identifikátor databáze). Rozdělení identifikátoru databáze a korespondujícího nástroje je následující:

    - databáze CCM – Change and Configuration Management
    - databáze RM – Change and Configuration Management
    - databáze DCC – Change and Configuration Management
    - databáze LQE – Change and Configuration Management

    Na stránce *Finalize Application* klikneme na tlačítko *Complete Application Setup* a po dokončení pokračujeme dále. Na stránce *Configure Data Warehouse* vložíme stejné nastavení jako v bodě 38. Obdobně postupujeme u nastavení všech ostatních nástrojů (Requirement Management a Data Collection Component).

40. U nástroje *Lifecycle Query Engine* nastavíme databázi, jako u předchozích nástrojů (použitý identifikátor je LQE). Na stránce konfigurace záloh, zakážeme zálohování a na poslední stránce dokončíme nastavení aplikace.

41. Nástroj *Tvůrce sestav* vyžaduje konfiguraci připojení na datový sklad nebo nástroj Lifecycle Query Engine. V tomto případě se připojíme k datovému skladu tak, že zadáme heslo databázového uživatele.

42. Na stránce *Finalize Lifecycle Project Administration Setup* klikneme na tlačítko *Finalize Application Setup* a poté přejdeme dále.

43. Poslední stránkou nastavení je souhrn nastavení, kde lze vidět veškeré nastavené položky. Pokračujeme tedy dále, kliknutím na tlačítko *Finish*. Další stránkou je administrace Jazz Team serveru, čímž se dá tedy instalace považovat za kompletní.

Instalace je časově náročná a složitá. Každý krok skýtá úskalí různých chyb nebo v budoucnu vzniklých návazných problémů, pokud tedy nastane problém, je nutné hledat pomoc na internetu. Případně bližší informace o nastalých výjimkách poskytne log nástroje JTS, který lze zobrazit pomocí příkazu `sudo pico /opt/IBM/JazzTeamServer/server/liberty/servers/clm/logs/jts.log`, kdy se vždy na konci souboru nachází nejnovější výjimky.

Kroky instalace se mohou v budoucích verzích od kroků zde popsaných lišit, proto je nutné maximálně využít jiných zdrojů a návodů, například rozcestníku instalace, který je dostupný zde.

Užitečné odkazy a zdroje: migrace databází, vytvoření databází, změna portu Jazz Team Serveru, instalace Jazz Team Serveru, odinstalace Jazz Team Serveru.

# C Návod k instalaci služby

Instalace služby je možná ve dvou režimech, prvním režimem je spuštění ve vývojovém prostředí a druhým režimem je nasazení do produkčního prostředí, tudíž například na vzdálený server.

## C.1 Spuštění ve vývojovém prostředí

Pro spuštění ve vývojovém prostředí není nutné mít nainstalované žádné speciální knihovny ani nástroje. Služba byla vytvořena jako konzolová aplikace, tudíž je pro její spuštění nutné mít nainstalované Visual Studio verze minimálně 2017 (nebo například Visual Studio Code s nutnými rozšířeními) spolu s .NET Core SDK verze alespoň 2.2.0.

## C.2 Nasazení na server

Nasazení na server bylo prováděno nad operačním systémem Ubuntu verze 18.04.01 LTS Bionic Beaver, tudíž i postup níže se bude vztahovat k tomuto operačnímu systému. Nasazení vyžaduje základní znalosti práce v prostředí operačního systému Linux a také uživatelská oprávnění na úrovni root. Předpokladem je také sestavená a spustitelná verze služby.

1. Stažení .NET Core SDK, dostupné zde, kde je nutné vybrat verzi dle použitého operačního systému (více informací zde).

2. Po stažení následuje vytvoření složky, ve které bude uloženo .NET Core SDK. To lze provést pomocí příkazu `sudo mkdir /bin/dotnet` (není vhodné vytvářet v domovské složce uživatele).

3. Dále je nutné stažený soubor rozbalit do cílové složky vytvořené v předchozím kroku příkazem `sudo tar zxf dotnet-sdk-2.2.203-linux-x64.tar.gz -C /bin/dotnet` (v tomto případě verze 2.2.203). Ověření dostupnosti .NET Core lze provést pomocí příkazu `/bin/dotnet/dotnet --version`, kdy výsledkem tohoto příkazu by měla být vypsána verze 2.2.203.

4. Tento bod je nutné provést pouze v případě, že prozatím není v operačním systému nainstalován software (démon) systemd, který spravuje služby v prostředí operačního systému Linux. Instalace příkazem `apt-get install -y systemd`.

5. Nyní je vhodné vytvořit uživatele, pod kterým se bude daná služba spouštět `useradd -m dotnetuser -p dotnetpass`, kde první parametr je uživatelské jméno a druhý heslo.

6. Vytvoření složky, ve které budou umístěny soubory služby, lze provést pomocí příkazu `sudo mkdir /etc/jazz-metrics`. Poté je nutné přenesený komprimovaný soubor služby případně přemístit do vytvořené složky `sudo mv publish.zip /etc/jazz-metrics` a rozbalit `sudo unzip publish.zip`.

7. Nyní přichází na řadu vytvoření samotné služby. Kroky jsou následující – přechod do složky démona systemd `cd /etc/systemd/system`. Druhým krokem je vytvoření konfiguračního souboru služby `sudo nano jazz-metrics.service`, který je nutné naplnit obsahem dle výpisu 9 (soubor se nachází v příloze *Service/jazz-metrics.service*, ten je tedy možné nakopírovat přímo) – pozor na korektní adresářovou strukturu v položkách nastavení WorkingDirectory a ExecStart.

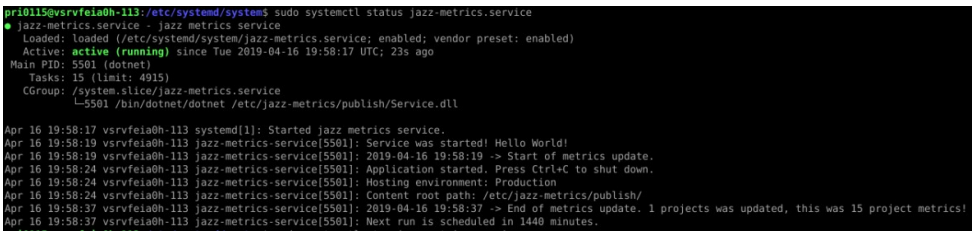
```
[Unit]
Description=Jazz metrics service for updating metrics.

[Service]
WorkingDirectory=/etc/jazz-metrics/publish
ExecStart=/bin/dotnet/dotnet /etc/jazz-metrics/publish/Service.dll
User=dotnetuser
Group=dotnetuser
Restart=on-failure
SyslogIdentifier=jazz-metrics-service
PrivateTemp=true

[Install]
WantedBy=multi-user.target
```

Výpis 9: Konfigurační soubor služby Jazz Metrics

8. Démona systemd je nutné znovu načíst pomocí příkazu `sudo systemctl daemon-reload`.
9. Povolení služby se provede příkazem `sudo systemctl enable jazz-metrics.service`.
10. Spuštění služby se provede příkazem `sudo systemctl start jazz-metrics.service`.
11. Zobrazení stavu služby se provede příkazem `sudo systemctl status jazz-metrics.service`. Korektní výsledek příkazu je zobrazen na obrázku 25.



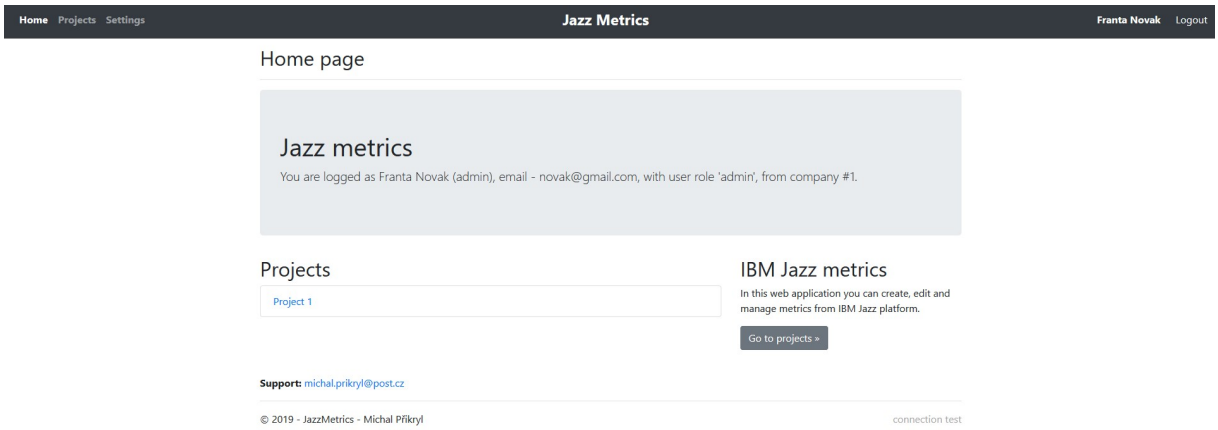

Obrázek 25: Korektní stav spuštěné služby Jazz Metrics

12. Případné ukončení služby lze provést příkazem `sudo systemctl stop jazz-metrics.service`.

# D Ukázka nástroje Jazz Metrics

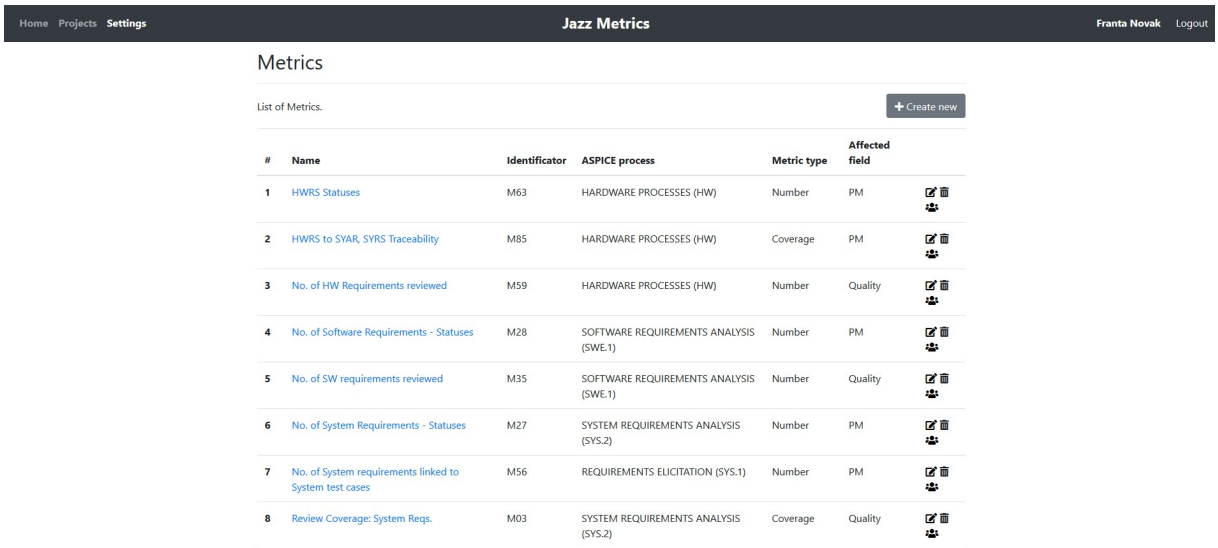
Na následujících snímcích lze vidět základní a nejdůležitější funkcionalitu vytvořeného nástroje s pracovním názvem Jazz Metrics. U každého obrázku se nachází popisek s vysvětlením, co daný obrázek zobrazuje a jaká základní funkcionalita je jím popsána/zobrazena.

- Na obrázku 26 je nástěnka nástroje – zobrazuje projekty, ve kterých participuje přihlášený uživatel. Dále zobrazuje také informace o přihlášeném uživateli.




Obrázek 26: Nástěnka nástroje Jazz Metrics

- Zobrazení dostupných metrik je možné pod položkou menu *Settings*, kliknutím na tlačítko *Go to setting* v oddílu *Metrics* (dostupné pouze uživatelé v rolích Admin a SuperAdmin). Metriky je možné editovat, mazat i vytvářet – viz obrázek 27.

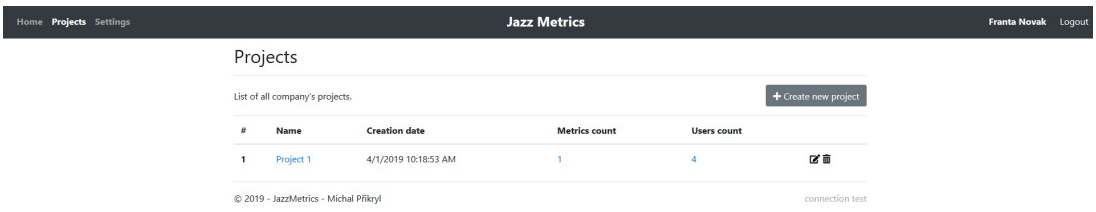



Obrázek 27: Seznam metrik dostupných uživateli v nástroji Jazz Metrics

- Vytvoření nové metriky je možné kliknutím na tlačítko *Create new* v seznamu metrik (viz obrázek 27). Po vyplnění potřebných parametrů a kliknutí na tlačítko *Create* je metrika vytvořena a dostupná k použití. Náležitou pozornost je nutné věnovat zadávání atributů metriky. Atributy metriky reflektují reálné atributy datového XML metriky, ze kterého se poté načítají data. Zobrazení nápovědy k jednotlivým možnostem atributu metriky je možné najetím myší na ikonu u daného textového pole. Ukázka formuláře na obrázku 28.

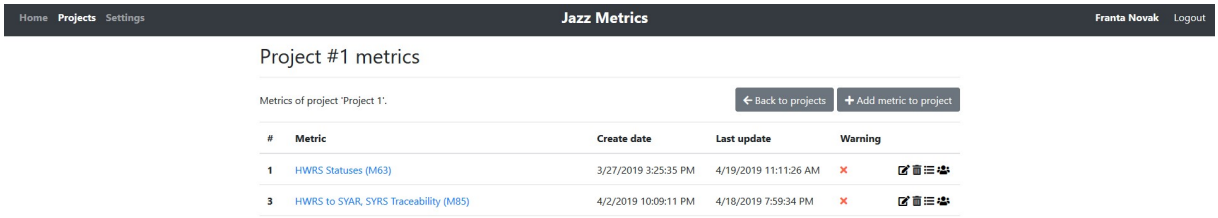

Obrázek 28: Vytvoření metriky v nástroji Jazz Metrics

- Zobrazení seznamu všech projektů, ve kterých přihlášený uživatel participuje je možné kliknutím na položku menu *Projects*, viz obrázek 29. Jednotlivé projekty je také možné upravit nebo smazat, dále upravovat přiřazené uživatele a metriky, kliknutím na číslo v odpovídající buňce tabulky.



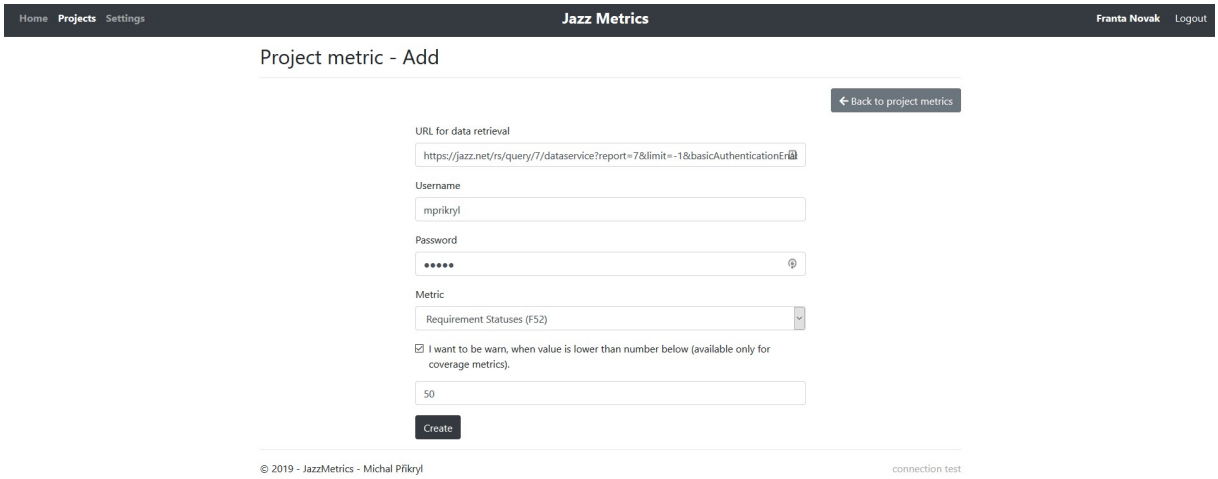

Obrázek 29: Seznam projektů uživatele v nástroji Jazz Metrics

- Kliknutím na číslo ve sloupci *Metrics count* (viz obrázek 29), lze zobrazit seznam projektových metrik vybraného projektu. Takovýto seznam projektových metrik se nachází na obrázku 30. Projektové metriky jsou takové, které jsou v rámci projektu sledovány a lze je editovat, mazat nebo zobrazit seznam s informacemi o aktualizaci dat (log).



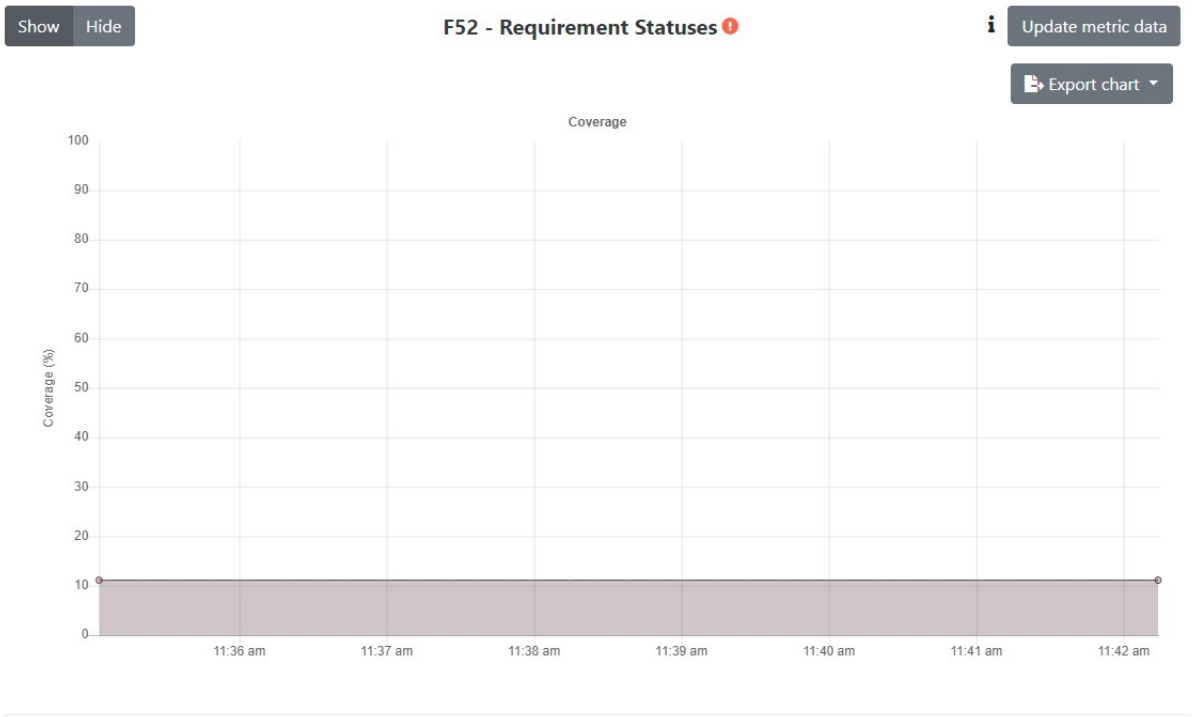

Obrázek 30: Projektových metrik na nástěnce projektu v nástroji Jazz Metrics

- Vytvořit novou projektovou metriku lze kliknutím na tlačítko *Add metric to project* (viz obrázek 30). Zde je při zadávání údajů nutné věnovat pozornost zadání správné URL adresy, ze které lze získat data nutné pro vybranou metriku a to tak, aby datový formát získaný z URL adresy odpovídal nastavení metriky (hlavně nastavení jednotlivých atributů metriky). Přihlašovací jméno a heslo jsou určené pro HTTP Basic autentizaci při stahování dat – tedy vytvořený účet v rámci IBM Jazz. Po zadání nezbytných údajů lze vytvořit projektovou metriku kliknutím na tlačítko *Create*. Ukázka vytvoření projektové metriky se nachází na obrázku 31.




Obrázek 31: Vytvoření projektové metriky v nástroji Jazz Metrics

- Kliknutím na název projektu v seznam projektů (viz obrázek 29), lze zobrazit nástěnku daného projektu se všemi projektovými metrikami. Na obrázku 31 vytvořenou projektovou metriku lze tedy po úspěšném vytvoření vidět na nástěnce, viz obrázek 32. Jelikož se jedná o metriku typu pokrytí (anglicky Coverage), nachází se zde jen jeden graf, který zobrazuje všechny stažené hodnoty metriky (momentálně pouze dvě). Informace o zobrazené metrice lze získat najetím myší na ikonu informace (*i*). Aktualizace dat metriky je možná kliknutím na tlačítko *Update metric data*. Export dat do formátů obrázku (png), dokumentu Microsoft Word (docx), sešitu Microsoft Excel (xlsx) nebo prezentace Microsoft PowerPoint (pptx) lze kliknutím na rozbalovací seznam *Export chart* a výběrem požadovaného formátu.




Obrázek 32: Seznam projektových metrik v nástroji Jazz Metrics

Při vytváření projektové metriky byla nastavena možnost upozornění na nízkou hodnotu metriky. Jelikož je aktuální hodnota metriky 10 % a zadaná prahová hodnota byla 50 %, je vedle názvu metriky zobrazeno upozornění na nízkou hodnotu. Taktéž je hlídán klesající trend metriky, který ale v tomto případě nenastal – metrika má stále stejnou hodnotu, čili neklesá.

# E Návod ke spuštění/nasazení nástroje Jazz Metrics

Spuštění vyvinutého nástroje je možné ve dvou režimech. Prvním režimem je spuštění na lokálním počítači, čili z vývojového prostředí. Druhým režimem je spuštění/nasazení pomocí programu Docker. Spuštění nástroje pomocí programu Docker je z pohledu počátečního nastavení značně jednodušší než spuštění z vývojového prostředí.

Nutností pro spuštění nástroje je v obou případech připojení daného počítače v sítí VŠB, ať už fyzicky nebo pomocí VPN a to kvůli dostupnosti databázového serveru. Druhou možností je vytvoření vlastní databáze (pomocí SQL skriptů umístěných ve složce *SQL*) a změna připojovacího řetězce v souboru *WebAPI/appsettings.json*.

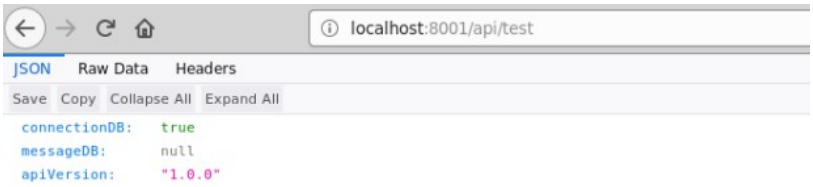
## E.1 Spuštění na lokálním počítači

Spuštění na lokálním počítači lze provést skrze vývojové prostředí, například Visual Studio verze minimálně 2017. Pro spuštění v módu „debug“ je nutné mít nainstalovanou vývojovou platformu .NET Core SDK verze 2.2.0, dále sadu funkcí Visual Studia s názvem *Vývoj pro ASP.NET a web*. Pro spuštění nástroje je nutné najednou spustit projekty WebAPI a WebApp, čímž se zajistí napojení prezentační vrstvy (WebApp) na aplikační vrstvu (WebAPI) a databázi.

## E.2 Nasazení na pomocí programu Docker

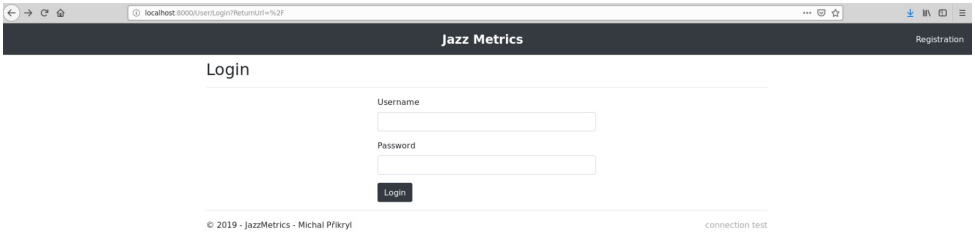
Nasazení pomocí programu Docker bylo prováděno nad operačním systémem Ubuntu verze 18.04.01 LTS Bionic Beaver, tudíž postup níže by měl být funkční pro operační systémy Linux (kontejnery jsou také nastaveny pro OS Linux). Nasazení vyžaduje základní znalosti práce v prostředí operačního systému Linux a také uživatelská oprávnění na úrovni root.

1. Vytvoření pracovní složky pomocí příkazu `sudo mkdir /etc/jazz-metrics`.
2. Nyní přejdeme do pracovní složky příkazem `cd /etc/jazz-metrics` a stáhneme repozitář se zdrojovými kódy nástroje pomocí příkazu `sudo git clone https://github.com/michalprikryl/jazz-metrics.git` (samozřejmě lze také použít zdrojové kódy z přílohy této práce).
3. Přejdeme do vytvořené pracovní složky příkazem `cd jazz-metrics/JazzMetrics`.
4. Nyní dochází k použití Dockeru, ten lze nainstalovat pomocí návodu zde. Sestavení kontejneru aplikační vrstvy lze provést příkazem `sudo docker build -f DockerfileWebAPI -t webapi .` (včetně tečky na konci příkazu), v případě nastalé chyby *Unable to load the service index for source https://api.nuget.org/v3/index.json.* přidejte k příkazu parametr *--network=host* (případně další řešení dostupné zde). Kontejner se poté spustí příkazem `sudo docker run -d -p 8001:80 webapi`. Úspěšné spuštění lze ověřit zadáním následující URL adresy v internetovém prohlížeči `http://localhost:8001/api/test`, korektní výsledek je zobrazen na obrázku 33.

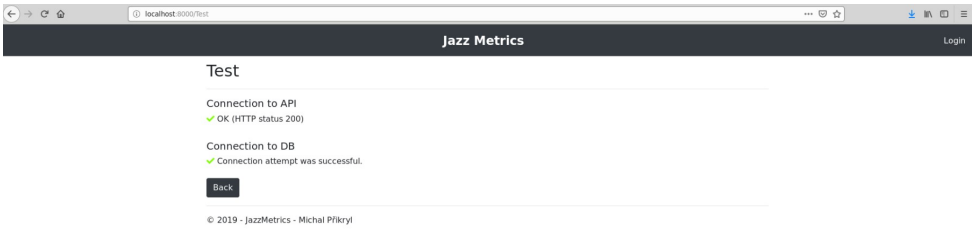



Obrázek 33: Korektní načtení aplikační vrstvy nástroje Jazz Metrics

5. Před samotným sestavením prezentační vrstvy je nutné změnit v souboru *Webapp/appsettings.json* IP adresu aplikační vrstvy změnou hodnoty *ServerApiUrl*, kde místo *localhost* náleží IP adresa hostujícího serveru. Sestavení kontejneru prezentační vrstvy nástroje lze provést příkazem `sudo docker build -f DockerfileWebApp -t webapp .` (včetně tečky na konci). Kontejner se poté spustí příkazem `sudo docker run -d -p 8000:80 webapp`. Korektní spuštění kontejneru lze ověřit zadáním URL adresy v internetovém prohlížeči `http://localhost:8000`, kdy by mělo dojít k načtení uživatelského rozhraní nástroje, konkrétně přihlašovací stránky, viz obrázek 34. Propojení prezentační a aplikační vrstvy lze otestovat kliknutím na odkaz *connection test* v dolní části uživatelského rozhraní. Úspěšný výsledek testu propojení aplikační a prezentační vrstvy je zobrazen na obrázku 35.




Obrázek 34: Korektní načtení prezentační vrstvy nástroje Jazz Metrics




Obrázek 35: Korektní propojení prezentační a aplikační vrstvy nástroje Jazz Metrics